In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Georgia State University, I agree that the Library of the University shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to quote from, to copy from, or to publish this thesis may be granted by the author or, in his absence, the professor under whose direction it was written, or in his absence, the Dean of the College. Such quoting, copying, or publication must be solely for scholary purposes and does not involve potential financial gain. It is understood that any copying from or publication of this thesis which involves potential financial gain will not be allowed without written permission of the author.

Jeff Wagon

Signature of Author

NOTICE TO BORROWERS

All dissertations and theses deposited in the Georgia State University Library must be used only in accordance with the stipulation prescribed by the author in the preceding statement.

The author of this thesis is:

Name: Jeff G. Wegrzyn

Street Address: 3220 Cape Circle

City, State, and Zip: Alpharetta, Georgia 30201

The director of this thesis is:

Professor: Dr. Frank O'Brien

Department: Chemistry

College: Arts and Sciences
Georgia State University
University Plaza
Atlanta, Georgia 30303-3088

Users of this thesis not regularly enrolled as students of Georgia State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their partrons are required to see that each user records here the information requested.

NAME OF USER	ADDRESS	DATE	TYPE OF USE

GAS CHROMATOGRAPH SIMULATOR FOR A COMMODORE 64 COMPUTER

A THESIS

Presented in Partial Fulfillment of Requirements for the
Degree of Master of Science
in the College of Arts and Sciences
Georgia State University

1984

by

Jeff Wegrzyn

Committee:

Frank E. Brown
Chairperson

H Fred Henneike
Member

Leland P. Vickers
Member

11/16/84
Date

Brian E. Armstrong
Graduate Director
College of Arts and Sciences

Clyde A. Gulp
Dean
College of Arts and Sciences

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES CONT.

LIST OF TABLES

I. INTRODUCTION

Gas-liquid chromatography is a common form of chemical analysis taught in many analytical chemistry courses. The operation of a gas chromatograph involves the balancing of variables, such as column choice, column temperature, and attenuation. Studying the effects that these variables have on a chromatogram can require many laboratory sessions, given the time required for reequilibration following changes of columns or column temperature. One new approach to teaching some of the more important aspects of analysis by GLC is the emulation of an instrument by interactive computer graphics (1). This thesis describes how gas chromatography theory has been combined with published chromatographic retention data to form a computer program that simulates the behavior of a basic GLC. The instructional objective of the simulation is to study the optimization of instrument parameters prior to a laboratory session. The simulation achieves this objective by permitting input of sample composition, column, oven temperature, flow rate, and attenuation. The resultant output is a chromatogram displayed on a high resolution graphics screen, with calculated solute specific retention volumes comparable to those obtained with laboratory instruments, operating under similar conditions.

This thesis begins with a brief section describing some

features of the computer language FORTH, which was used to write the simulation program. This is followed by a chapter on gas chromatography theory that forms the foundation for the simulator. In this section the equations necessary to predict retention times, and peak shapes are described. Also included are listings of the same equations translated into FORTH source code. In chapter IV the program's structure is explained, with emphasis placed on the FORTH words that control the 10 major simulator functions. To determine how well the program emulates laboratory instruments a number of experiments were performed on GOW-MAC, and Perkin-Elmer gas chromatographs. Similar experiments were performed on the simulator, and the results are compared in chapter V. The thesis is concluded by a listing of the entire simulator program.

II. FORTH

The gas chromatography simulation program is written in the computer language of FORTH. This language offers several distinct advantages over the more traditional microcomputer languages, such as BASIC. The most attractive feature of FORTH is that it can be extended (2). That is, if a function needs to be performed and is not included in the language it can be added. FORTH begins with a powerful set of predefined commands called words. These words are collectively stored in FORTH's word set, its dictionary. Programs are constructed by taking the words from the dictionary, and combining them into a new word. If the sequence of functions called out by the new word is acceptable, it becomes part of the dictionary, and can be used in any future words. This building block approach is FORTH's equivalent of high-level coding.

When optimally used, FORTH can be very descriptive, with each new word describing the function it performs. A simple example would be:

```
: CLEAR 147 EMIT ;
```

The colon creates a new dictionary entry by the name of CLEAR. The words following the name make up the new word's source code, or definition. In this case the definition consists of just one number, and one word called EMIT. The definition is concluded by the semi-colon. After CLEAR is

compiled into the dictionary, it can be executed by typing "CLEAR", or placing it in the source code of a new word. When it is executed, the FORTH interpreter looks up the source code for CLEAR. The first string of characters it encounters is the code for 147. Since this is not a word in the dictionary, the interpreter checks to see if it is a number. It is, so it is stored as a number on a stack. The definition for EMIT is then executed. This word takes the top number on the stack and transmits it as an ASCII character to the selected output device. On the Commodore 64, the ASCII code for 147 will clear the screen and place the cursor in the upper left corner. Therefore the word CLEAR does exactly what it says: it clears the screen.

FORTH compiled code is compact, with applications requiring less memory than equivalent assembly-language programs (3). This compiled code executes quickly, with speeds approaching 80% of that of machine code. The combination of these features makes FORTH a very powerful language, which gives the simulator many features not possible with most high-level microcomputer languages.

The FORTH package used to create the gas chromatography simulator was purchased from Performance Micro Products. This version contains floating-point mathematics, and a number of graphics words, along with the FORTH-79 standard dictionary.

III. GAS CHROMATOGRAPHY THEORY

GAUSSIAN CURVES

In a gas chromatographic separation, a sample containing the solutes is injected onto a heated block, where it is immediately vaporized. The carrier gas stream sweeps it into the column as a plug of vapor . In the column, the solutes are absorbed by the liquid phase and then desorbed by fresh carrier gas. This partitioning process can occur hundreds to thousands of times as the sample is moved toward the outlet by the carrier gas. As the solute undergoes all these phase changes, the original plug spreads out. By the time the solute is eluted, the zone containing it can be described as a Gaussian distribution, and will be recorded as a peak by the detector. This type of distribution can be described mathematically by the expression:

$$y = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x - u)^2}{2\sigma^2} \qquad (1)$$

In this equation x is an independent variable that represents values of individual measurements, and u is the arithmetic mean of the measurements. The quantity (x - u) is the deviation from the mean, and y is equal to the number of results for each value of (x - u). The symbol σ represents the standard deviation of the distribution. In a Gaussian distribution or curve, σ is directly related to the breadth of the curve.

Equation 1 can be simplified by introducing the variable z for the exponential term.

$$z = \frac{x - u}{\sigma} \qquad (2)$$

z gives the deviation from the mean in units of standard deviation. Combining equations 1 and 2 will give the expression:

$$y = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(z)^2}{2}\right] \qquad (3)$$

Any curve calculated by using equation 3 will have 68.3 % of its total area lying within one standard deviation ($\pm 1\sigma$) of the mean; 95.5 % will be within $\pm 2\sigma$, and 99.7 % will be within $\pm 3\sigma$ (4). This property is useful when developing an algorithm that generates Gaussian curves. If 68.3 % of the curve area exists between plus and minus one standard deviation, then the height of the curve at $\pm\sigma$ must be a specific percentage of the peak height at the mean. For example Table 1 gives the y values, calculated from equation 3 for a series of σ and z values. It can be seen from this data that the ratio of peak height to peak maximum at various z values is not affected by the σ value.

In the simulator, Gaussian curves are used to represent solute peaks in chromatograms. These peaks will be plotted on a high resolution screen containing 320 x 200 pixels.

σ	z	y	$\frac{y}{y_{max}}$
1	0	.3989	1
1	1	.2420	.6065
1	2	.0540	.1353
2	0	.1995	1
2	1	.1210	.6065
2	2	.0270	.1353
3	0	.1330	1
3	1	.0807	.6065
3	2	.0180	.1353

Table 1. Ratios of peak height to peak maximum. $y_{max} = y$ @ $z=0$

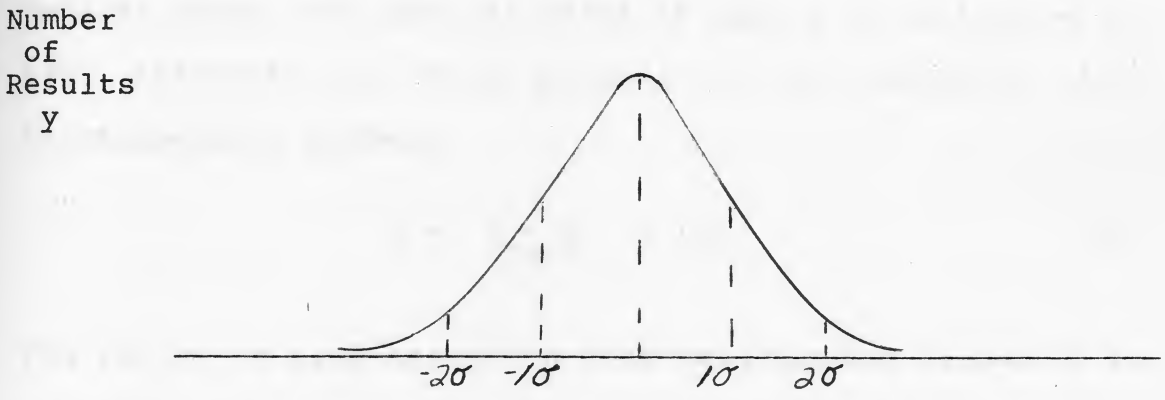

Figure 1. Graphical representation of a Gaussian distribution.

The x coordinate of each pixel will represent a specific time. For example, if the width of the screen represents 5 minutes, then each pixel will represent 0.94 seconds of the total time. This enables the program to convert a solute retention time into an exact pixel coordinate. Therefore if the retention time and standard deviation for a

distribution are known, the points of a Gaussian curve can be easily generated and stored. This is done by substituting the program variables U and S along with a DO-LOOP index I, into equation 2.

$$z = \frac{I - U}{S} \qquad (4)$$

The variable U holds the retention time, S contains the standard deviation value, and I represents the x coordinate of the high-res pixels. When values of 0 through 320 are placed in I, the z value for each x coordinate can be determined by equation 4. To keep the mathematics in the integer range, the absolute value of each z is multiplied by 100. Integers run more quickly in the computer than floating-point numbers.

$$z = \frac{I - U}{S} \times 100 \qquad (5)$$

The ratios of peak height to peak maximum for values of z, ranging from .1 to 3, are sequentially stored in the Gaussian Curve buffer in increments of .01. To save memory,the ratios are multiplied by 1000 and stored as 16-bit integers. A listing of the buffer is presented on SCR# 89 in the program listing. The calculated z value is used to recall from storage the proper ratio for each x coordinate. This is done by placing each z value into equation 6.

$$\text{memory location} = 2z - 20 + 51300 \qquad (6)$$

For example, if z from equation 5 equals 100, then equation 6 will equal 51480, which is the memory location where 607 is stored. This value is equal to the peak height to peak maximum ratio, multiplied by 1000, at 1 standard deviation unit away from the curve mean. The y coordinate of the pixel on the curve is determined by multiplying 607 by the maximum peak height, and dividing the result by 1000. Calculation of y coordinates for all x coordinates between 0 and 320, defines the entire curve. The coordinates of the points are stored in a buffer called PB (Point Buffer). The y values are stored sequentially according to increasing x values. The x values can be determined from their memory locations. the memory locations of a buffer can represent the pixel x coordinate, each y value is stored sequentially, in a buffer called PB (POINT BUFFER).

The FORTH words that generate and store points for Gaussian curves are POINTS, DP1, and DP2, (Data Points 1 & 2). The simulator's chromatograms are displayed on a horizontally split screen with DP1 generating the top half, and DP2 controlling the bottom portion.

```
: POINTS 2* 20 - GAC + @ H @ 1000 */ ;
: DP1 320 0 DO
    I INTFP U FP@ FP- FPABS
    S FP@ FP/ FP 100 FP* FPCK FPINT
    DUP 10 < IF DROP H @ PB @ I 2* +
    HGHTCK ELSE DUP
```

```
300 > IF DROP ELSE POINTS
PB @ I 2* + HGHTCK THEN THEN LOOP ;
```

DP2 is the same as DP1 except the loop index is set from 320 to 640.

The second and third lines in DP1 are equivalent to equation 5, with the calculated z value being left on the stack. This number is then checked. If it is less than 10, the current I value is at the peak maximum. If the z value is greater than 300, the I value is more than 3 standard deviation units away from the peak maximum, and the point becomes part of the baseline. When the z value falls between 10 and 300, the word POINTS is executed, performing the operation described in equation 6.

Additional non-standard FORTH words that are in DP1 and POINTS are:

FPCK - Checks the z value, so it does not fall outside a designated range.

HGHTCK - Ensures that the calculated y pixel coordinate does not exceed 83. The maximum number of vertical pixels available on a single chromatogram screen is 83.

Most of the words discussed in this section are listed on SCR# 23 in the program listing.

RETENTION

The uncorrected retention volume V_R is the volume of carrier gas required to elute a compound from the column. Normally this is measured on a chromatogram as a function of distance from the point of injection to the peak maximum. Under constant pressure conditions, the flow rate is linear with time. Therefore retention volume can also be expressed as retention time t_R. The V_R and t_R are characteristic of a compound and the column liquid phase, so they can be used to identify the compound (5). The retention volume is not influenced by the presence of other compounds within a sample.

The retention volume is dependent on many contributing factors that are unique to the column on which the separation takes place. Within each packed column there exists free space between the individual particles of the packing material. During instrument operation this space is occupied by the mobile phase V_m . This volume can be determined for any packed column by the expression:

$$V_m = T_m F_o \qquad (7)$$

T_m - Time, in minutes from injection to peak maximum of an unabsorbed gas.

F_o - Flow rate, calculated from the observed flow rate F, in ml/min, at column-exit temperature,

pressure, and corrected for the vapor pressure of water (6).

$$F_o = F \left(\frac{P_a - P_w}{P_a} \right) \frac{T}{T_a} \qquad (8)$$

P_a - Pressure at which flow was measured, in mm of mercury.

P_w - Vapor pressure of water, in mm of mercury, at temperature of flow rate measurement.

The uncorrected retention volume can be expressed as:

$$V_R = t_R F_o \qquad (9)$$

The value of retention volume is dependent on the size of the mobile phase V_m. To eliminate this dependence, a term called adjusted retention volume V'_R is introduced.

$$V'_R = V_R - V_m \qquad (10)$$

Equation 10 gives the retention volume of an unabsorbed gas. When V'_R is corrected for gas compressibility it becomes the net retention volume V_n.

$$V_n = j\, V'_R \qquad (11)$$

j - Pressure gradient factor of Martin and James (7).

$$j = \frac{3}{2} \times \frac{(P_i/P_o)^2 - 1}{(P_i/P_o)^3 - 1} \qquad (12)$$

P_i - Column inlet pressure.

P_o - Column outlet pressure.

Net retention volume is dependent on the amount of liquid phase coated onto the column packing. To negate this variable, chromatographic results can be reported by a method devised by Littlewood, Phillips, and Price, called specific retention volume, V_g (8). The specific retention volume corresponds to the volume of gas at 0^oC required to elute one-half of the solute from a column which contains one gram of liquid phase, and has no pressure drop or free gas space (9).

$$V_g = \frac{V_n \times 273}{W \times T} \qquad (13)$$

T - Temperature of column in degrees Kelvin.

W - Weight, in grams of liquid phase in column.

V_n - Net retention volume.

The temperature dependence of a specific retention volume is given by the relation in equation 14 (10).

$$\log V_g = \Delta H/2.3RT + \text{const} \qquad (14)$$

ΔH - Partial molar heat of solution of solute in liquid state.

R - Gas constant.

T - Absolute temperature.

For many systems, plotting $\log V_g$ against $1/T$ for several different isothermal runs will result in an approximately linear function. Over a limited temperature range the

resultant line can be expressed by an Antoine equation:

$$\log V_g = B(T + C) + A \qquad (15)$$

A - Constant, equal to the y axis intercept.

B - Constant, slope of the resulting line and is equal to $\Delta H/2.3R$.

C - Constant, reduces to 273 for systems that follow equation 14.

An example of a system that follows equation 15 appears in figure 2. In that graph the log V_g has been plotted against 1/T for ethanol on a Carbowax 20M column. The data for this graph was taken from McReynolds book on Gas Chromatographic Retention Data (11). Using the plots of experimental data, it is possible to determine the constants A and B for a solute on a liquid phase. These constants have been calculated for all 16 solutes, on the 4 liquid phases available in the simulation. The values are stored in the STCKRM (Stockroom) buffer.

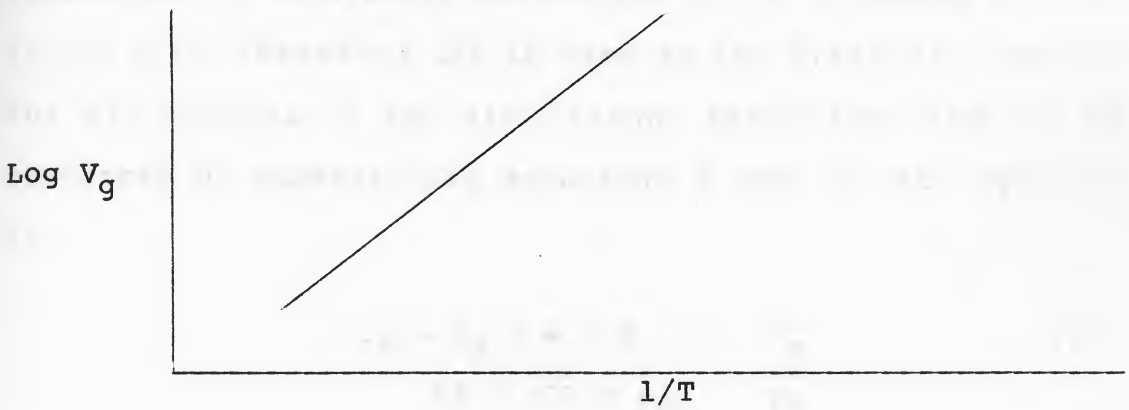

Figure 2. Log V_g versus 1/T for ethanol on a Carbowax 20M column.

Whenever the simulation is required to predict a retention time, the program steps backward through equations 15, and 13 - 9. First, the constants A and B are recalled from the STCKRM buffer, and used to calculate log V_g by equation 15. The inverse log is calculated, leaving the specific retention volume on the stack. The adjusted retention volume can be determined by rearranging equation 13 and substituting the expression in equation 12 for net retention volume.

$$V'_R = \frac{V_g \times W \times T}{j \times 273} \qquad (16)$$

Within the FORTH word that performs equation 16, W is a variable that contains the weight, in grams, of the liquid phase. The weight of liquid phase will be different for each of the 4 columns in the simulation. Another variable, labled T, holds the column absolute temperature. From the literature, a reasonable estimation for j in packed columns is .65 (10). Therefore .65 is used as the pressure gradient for all columns in the simulation. Retention time can be predicted by substituting equations 9 and 10 into equation 16.

$$t_R = \frac{V_g \times W \times T}{.65 \times 273 \times F_o} + \frac{V_m}{F_o} \qquad (17)$$

The simulation assumes the carrier gas flow rates are

measured on a soap-bubble flow meter that is at 25°C and under 1 atmosphere pressure. The vapor pressure of water at 25°C is 24 mm Hg. Use of this constant in equation 18, will determine the F_o value for any temperature and flow rate.

$$F_o = \frac{F \times T}{298} \times \frac{736}{760} \qquad (18)$$

F - Observed flow rate.

T - Absolute column temperature.

The last term to be studied in equation 17 is the mobile phase volume V_m. This value depends on the type and mesh size of the support material. In order to estimate the inter-particle space available to the mobile phase, the retention time of air was recorded for 12 runs on a Carbowax 20M column. The data are presented in table 2. From these results it was determined that a 4 ft. column has about 21 mL of space available to the mobile phase.

Column: Carbowax 20M
Dimensions: Length - 4 ft, .25" O.D., .210"I.D.
Column Temp: 110°C
Flow Rate: 50 mL/min

t_R =	.41 min	.50 min
	.53	.55
	.52	.49
	.48	.53
	.58	.52
	.51	.54
	average t_R = .51 min	
	V_m = 20.96 ml	

Table 2. Retention time of an unabsorbed gas on a Carbowax 20M column.

With the estimates for j and V_m, the simulation program can predict retention times by the following equation:

$$t_R = \frac{V_g \times W \times T}{.65 \times 273 \times F_o} + \frac{20.96}{F_o} \qquad (19)$$

Equation 19 is handled in FORTH by the words AIR and RETTIME.

```
: AIR FL FP@ T FP@ FP* FP .00211 FP*
    FL3 FP! ;
: RETTIME FP 10 B FP@ T FP@ FP/ A FP@
    FP+ FPI W FP@ T FP@ FP* FP*
    FP 273 FP/ FP .65 FP/ FP 20.96 FP+
    FL3 FP@ FP/ FP 64 FP* U FP! ;
```

AIR solves equation 18, and stores the result in the variable FL3. In RETTIME, the first line calculates V_g for a given temperature. The remaining FORTH source code in RETTIME solves equation 19, and leaves the calculated retention time on the stack. The stack is multiplied by 64, converting it to an x coordinate pixel location. It is then stored in the variable U, where it is used by DP1 and DP2 as the mean for a Gaussian curve.

PEAK WIDTH

On a typical chromatogram, early peaks are relatively high and narrow, while late peaks are low and broad. To replicate this type of behavior accurately, portions of plate theory and rate theory are incorporated into the simulator program.

In plate theory as developed by Martin and Synge, a column is described as a series of discrete,narrow, horizontal layers called theoretical plates (13). Equilibration of the solute between mobil and liquid phases it is assumed to take place in each plate. The efficiency of a column can be evaluated by calculating the number of theoretical plates N. Column efficiency improves as the number of theoretical plates increases. One method for determining plate number is to measure the peak width y, and the distance from injection to peak maximum x, on a chromatogram. These values are then placed into equation 20, which calculates N.

$$N = 16 \left(\frac{x}{y}\right)^2 \qquad (20)$$

Plate theory is useful for determining column efficiency, but it fails to explain peak shape adequately. For example, there is no mechanism to account for the effects of flow rate and packing characteristics on zone broadening.

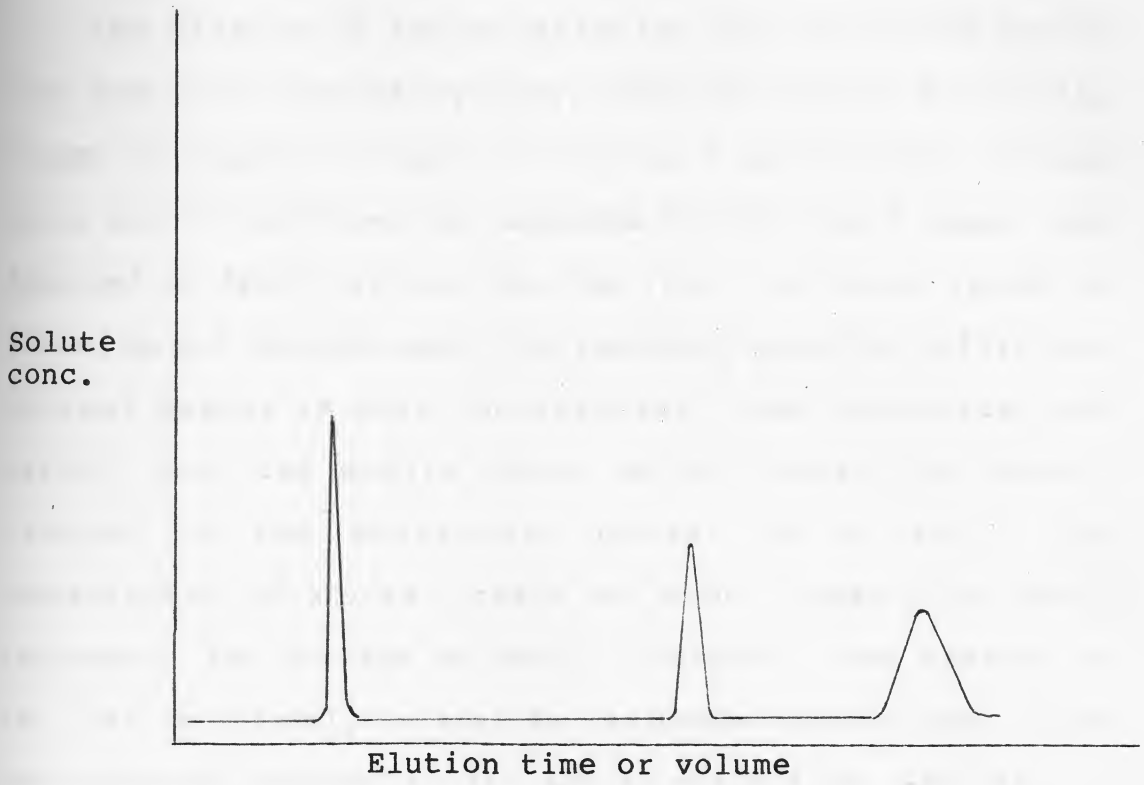

Figure 3. Typical gas chromatographic separation.

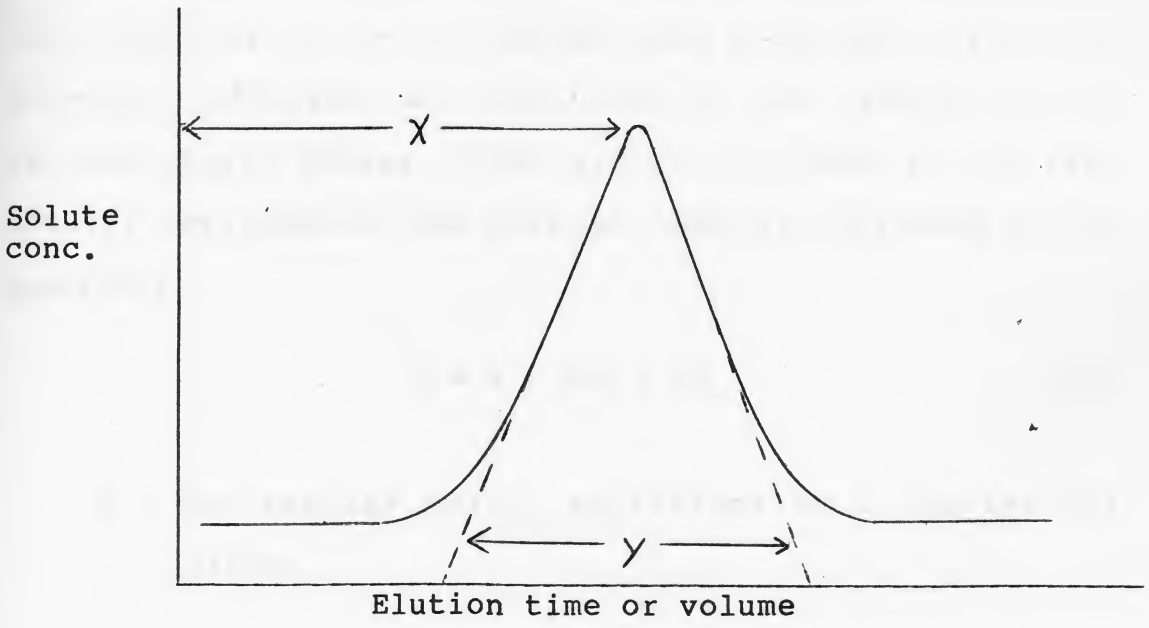

Figure 4. Calculation of theoretical plates.

The effects of these variables are explained in the rate theory of chromatography. When the solute enters the column as a plug of vapor, it begins a partitioning process where solute molecules are absorbed by the liquid phase, and desorbed by fresh carrier gas. The time a molecule spends in either phase depends upon its randomly gaining sufficient thermal energy to make the transfer. Some molecules move quickly into the mobile phase, while others lag behind, trapped in the stationary phase. As a result the concentration of solute spreads out about a mean value which represents the average molecular behavior. The breadth of the zone is directly related to residence time in the column and inversely related to the mobile phase flow rate (14).

The migration process just described does not account for the entire width of the solute zone. Additional contributions to the broadening come from eddy diffusion, molecular diffusion, and resistance to mass transfer in the gas and liquid phases. These are all included in the rate theory, developed by van Deemter, and are expressed as the equation:

$$H = A + B/\mu + C\mu \qquad (21)$$

H - The average height equivalent to a theoretical plate.

$$H = L/N \qquad (22)$$

Where L is column length, usually in centimeters, and N is the number of theoretical plates.

A - Constant, related to particle size and multiplicity of gas paths (eddy diffusion).

μ - Linear gas velocity

$$\mu = \frac{\text{length of column (cm)}}{\text{retention time of air (sec)}} \qquad (23)$$

B/μ - Constant, related to diffusion coefficient of the mobile phase.

$C\mu$ - Constant, represents resistance to mass transfer in the gas and liquid phase.

Plotting H versus μ will result in a hyperbolic curve. The minimum in the curve occurs at a flow rate at which the column operates most efficiently.

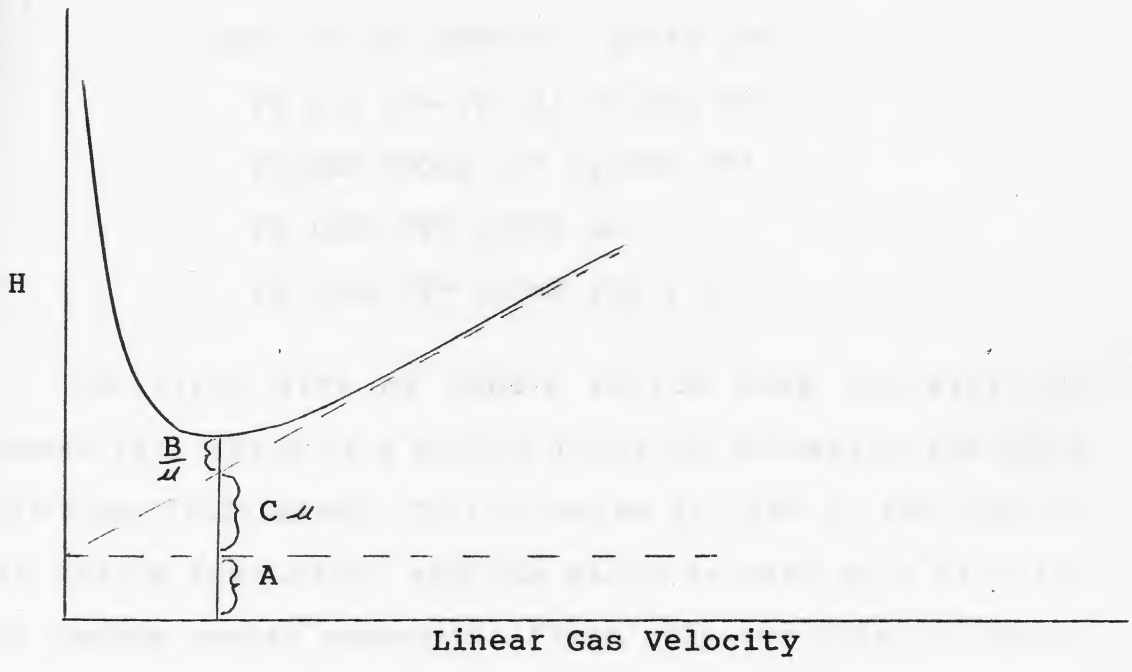

Figure 5. Plot of H versus μ .

Rate theory postulates 5 interdependent terms that influence band width. These are flow rates, solute residence time, eddy diffusion, longitudinal diffusion, and resistance to mass transfer. The last three terms are dependent upon how a column is constructed, so their values differ with each column. The simulation program accounts for this by using a random number generator to create the B and C terms in the van Deemter equation. This takes place at the beginning of the program, when a column set number is assigned. The number is actually a code where the first 3 digits represent the van Deemter B term, and the last 2 digits represent the C term. The A term will be a constant since all the columns are packed with the same size support material. The FORTH word CODE, generates the B and C terms.

```
: CODE 162 C@ INTFP FP .00196 FP*
     FP 2.5 FP+ FP .15 FPOVER FP/
     FPSWAP FPDUP FP* FPOVER FP*
     FP 1000 FP* FPINT N2 !
     FP 1000 FP* FPINT FL2 ! ;
```

The first line of CODE's source code contains the number 162, which is a memory location accessing the C64's real-time jiffy clock. This location is read at the time of the word's execution, and its value is used as a seed for the random number generator. Since only one byte of memory is read, the seed can range from 0 to 255. The remainder of

the word converts the seed into the B and C terms of the van Deemter equation, and stores them in the variables N2 and FL2. Use of the random number generator makes possible a large number of columns with varying degrees of efficiency. The program also has an option to enter a previously generated column set number, permitting the operator to turn the computer off and still use the same set of columns at a later time.

When prediction of a peak width is required, the program recalls the B and C terms generated by CODE, and calculates the value of H. The FORTH word that performs this operation is VAND.

```
: VAND FP 122 FPDUP FP 20.96
    FL3 FP@ FP/ FP 60 FP* FP/ MU FP!
    FP .08 N2 @ INTFP FP 1000 FP/ MU
    FP@ FP/ FP+ FL2 @ INTFP FP 1000
    FP/ MU FP@ FP* FP+ FP/ FP 2.5 FP*
    N FP! ;
```

This word is a combination of two equations. Lines 1 and 2 calculate the linear gas velocity by equation 23, using the current value for flow rate. The remainder of the word converts H to the number of theoretical plates. This is done by rearranging equation 22.

$$N = L / H \qquad (24)$$

L - Length of column in centimeters (122 for a 4 ft column).

The result of this equation is stored in the floating point variable N .

Prior to the execution of VAND, the residence time of the solute is calculated by the word RETTIME, and stored in the variable U. Therefore, when VAND calculates the number of theoretical plates, all terms necessary for peak width determination by equation 20 are stored in program variables.

$$y = \sqrt{\frac{N}{16}} \; x \qquad (25)$$

Equation 25 is translated into FORTH by the word SIGMA.

```
: SIGMA U FP@ N FP@ FP 16 FP/ FPSQR FP/
        FP 4 FP/ S FP! ;
```

A peak width determined by equation 25 is equal to 4 standard deviation units of a Gaussian distribution. Therefore SIGMA divides the product of equation 25 by 4, and stores the result in the variable S. This number is used by the Gaussian curve words DP1 and DP2 as the standard deviation value when peak shape is calculated.

PEAK HEIGHT

The concentration of solute leaving a column can be expressed by equation 26 (15).

$$C = (N^{1/2}/V^{O}_{R})\ [\ W/(2\pi)^{1/2}\ ]\ \exp\ -\ (N/2)(1 - V^{O}/V^{O}_{R}) \qquad (26)$$

V^{O} - Volume of gas entering or leaving a column. The gas phase is assumed incompressible.

V^{O}_{R} - Corrected retention volume of solute.The mobile phase is assumed incompressible.

N - Number of theoretical plates.

W - Initial weight of solute.

C - Concentration of solute.

When $V^{O} = V^{O}_{R}$, the exponential term in equation 26 will equal 1, leaving only the first half of the equation. Solving for C under these conditions will give the concentration maximum.

$$C_{max} = (N^{1/2}/V^{O}_{R}\ )\ W/(2\pi)^{1/2} \qquad (27)$$

This equation is used by the program to calculate the maximum peak height. Information needed in equation 27 is obtained from the STOCKROOM screen. This screen lists the 16 chemicals that are available for sample preparation. A chemical is placed into a sample by moving the cursor to the appropriate row and column, and entering the volume. Any amount between 0 and 99 milliliters can be entered or

changed while the screen is displayed. The volumes are stored in the STOCKROOM buffer, which has its beginning address held in the variable STCKRM. These operations can be compared to going through a stockroom and pipetting chemicals into a single beaker. When the operator exits the screen, the total sample volume is determined, and the percentage of each component is calculated. Those values are stored in the STOCKROOM buffer. The percentages will be passed along to the variable H when a peak height is calculated.

The word that duplicates equation 27 is called AREA.

```
: AREA N FP@ FPSQR U FP@ FP 64 FP/ FP/
    H @ INTFP INTFP FP 25.07 FP/ FP* FP .5
    FP+ FPINT H ! ;
```

In the first line of AREA, the number of theoretical plates N is recalled and the square root determined. The retention time, stored in U, is fetched and converted from a pixel x coordinate back to minutes. The variable H contains the percentage of total sample volume for a particular component. The number 25.07 is equal to $(2\pi)^{1/3}$ multiplied by 10. This magnification is used to place the calculated peak heights in the range of the high resolution screen's y coordinate. The value obtained from equation 27 is rounded off to the closest integer, and stored back in the variable H.

IV. PROGRAM DESIGN

The gas chromatography simulator is a turnkey program. This means that immediately following the initial program loading, the last word in the FORTH dictionary is executed. Since FORTH uses a building block style of programming, it is not difficult to operate an entire program from one word. The last word in the simulator dictionary is MENU.

```
: MENU BLOAD STARTUP COLUMNS BEGIN
        MENDSLY KEYIN DUP
        133 = IF DROP DEMO 0 ELSE DUP
        137 = IF DROP SAMPLE 0 ELSE DUP
        134 = IF DROP INST 0 ELSE DUP
        138 = IF DROP GC 0 ELSE DUP
        135 = IF DROP PRV 0 ELSE DUP
        139 = IF DROP HETP 0 ELSE DUP
        136 = IF DROP UNKNOWN 0 ELSE DUP
        140 = IF DROP PCMPR 0 ELSE DUP
        144 = IF DROP SMPLCLR 0 ELSE
        5 = IF NEW 0 ELSE 0 THEN THEN THEN
        THEN THEN THEN THEN THEN THEN THEN
        UNTIL ;
```

MENU begins by executing BLOAD (Block Load), which fills the buffer memory with 10 screens. These screens are not compiled into the dictionary, but merely copied into RAM memory. Contained within the screens are the messages and data necessary to run the simulation. By having this

information in RAM, word execution is very quick, with no time lost due to accessing disk storage. Following BLOAD is STARTUP, which fills key variables with initial values. The variables contain numbers greater than zero, to avoid the system crash that would occur if in a mathematical operation we attempted to divide by zero.

The next word to be executed is COLUMNS, which assigns a set of 4 columns by generating a column set number, or permits entry of a previously generated column set. Details of COLUMN's source code is described in the section on PEAK WIDTH. The program now has all the variables filled, and buffer information necessary for operation, and is ready for keyboard input.

The FORTH-79 word BEGIN marks the start of a DO-LOOP, it also serves as a return point from the corresponding UNTIL. MENU is programmed so the conditions for exiting the loop can never be met, so the simulator is contained within an infinite loop always returning to BEGIN. Just inside the loop MENDSPLY clears the monitor screen and transfers the simulator's menu from the B4 block buffer to the monitor (see figure 6). The word KEYIN suspends program execution and awaits input from the keyboard. Once a key is struck KEYIN will leave the ASCII value of that key on the stack. The number is immediately duplicated by DUP, and compared to the column of numbers in MENU. If a match is found the program will drop the extra ASCII value, and execute the

```
****** GAS CHROMATOGRAPH SIMULATOR *****

                MENU

     (F1) INTRODUCTION AND DEMONSTRATION

     (F2) GO TO GC STOCKROOM

     (F3) INSTRUMENT CONTROLS

     (F4) GC INSTRUMENT SCAN

     (F5) DISPLAY PREVIOUS SCREEN

     (F6) HETP EXPERIMENT

     (F7) UNKNOWN EXPERIMENT

     (F8) CHECK UNKNOWN ANSWER

  (CTRL 1) CLEAR UNKNOWN OR SAMPLE

  (CTRL 2) LEAVE THE SIMULATOR

     SELECT THE DESIRED FUNCTION BY

     PRESSING THE DESIGNATED KEY !

-------------------------------------------------------
```

Figure 6. Listing of the Gas Chromatograph Simulator menu.

word between IF and ELSE. For example, if the operator pressed F3, KEYIN would place the ASCII code of 134 on the stack, thereby executing INST. The word INST displays the instrument settings, and accesses the words that allow changing of the settings. Following INST is a 0, which tells UNTIL to return the program pointers back to BEGIN. If the ASCII value left on the stack by KEYIN cannot be matched, the number is erased and the menu redisplayed.

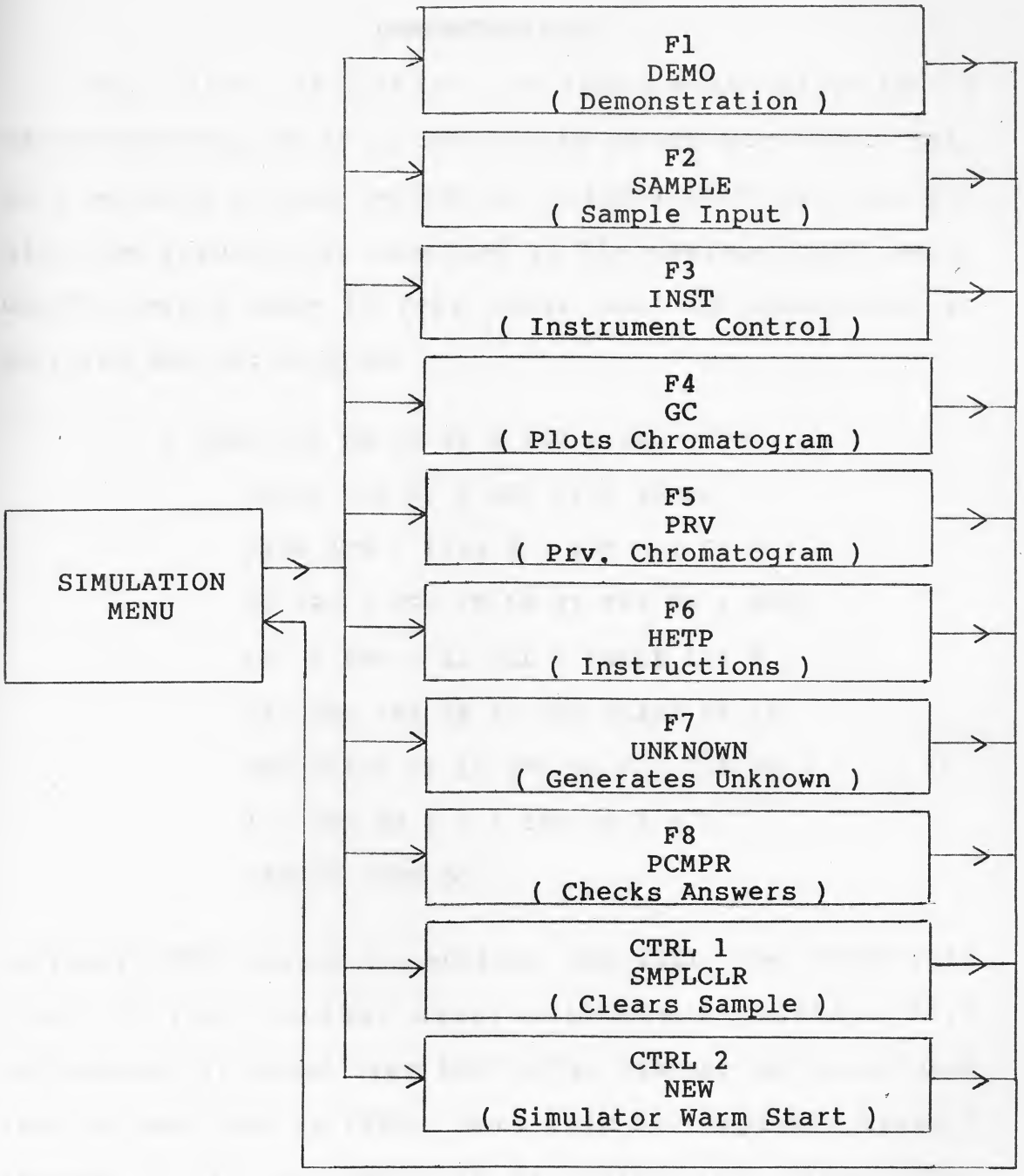

Figure 7. List of FORTH words executed by menu selections.

DEMONSTRATION

The first selection on the menue display is DEMONSTRATION, which is controlled by the word DEMO. This word performs a trial separation to familiarize the operator with the simulation. Compared to the average FORTH word, DEMO's source code is very long, but the operations it performs are not complex.

```
: DEMO CLR CR CR B1 @ 360 + 640 TYPE
     PAUSE CLR B7 @ 880 TYPE PAUSE
     1246 ATN ! 1764 R ! 120 O ! 50 F !
     FP 393 T FP! FP 50 FL FP! FP 1 ATN'
     FP! 0 RNG ! 21 COL ! DSCLR 161 @
     151 AND 162 C@ 15 AND 56234 C@ 15
     AND 56325 C@ 15 AND DS @ ! 116 DS @
     + ! 232 DS @ + ! 248 DS @ + !
     SAMPLE INST GC ;
```

Initially DEMO clears the monitor, and fills the screen with a set of instructions describing the simulator. This information is moved from RAM buffer storage and translated into screen code by TYPE. Once this is complete, lines 3 through 5 fill a number of variables with preselected values. Next a simple random number generator is used to prepare a sample containing different volumes of 4 alcohols. The generator receives its seed value from the real-time jiffy clock, and limits the volume range of each alcohol

to 0 to 15 milliliters. The word SAMPLE then displays the selected volumes on the stockroom screen. At this time all input and display operations for sample preparation are active, so the sample composition can be changed if desired. Pressing the RUN/STOP key will display the final sample composition. INST is then executed, displaying the instrument contol settings. When program execution is continued, the word GC calculates peak shape and retention time, and switches to high-resolution graphics to plot the chromatogram. The program then loops back to the menu. The experimental conditions are not optimized, presenting a good reason to experiment with the instrument settings.

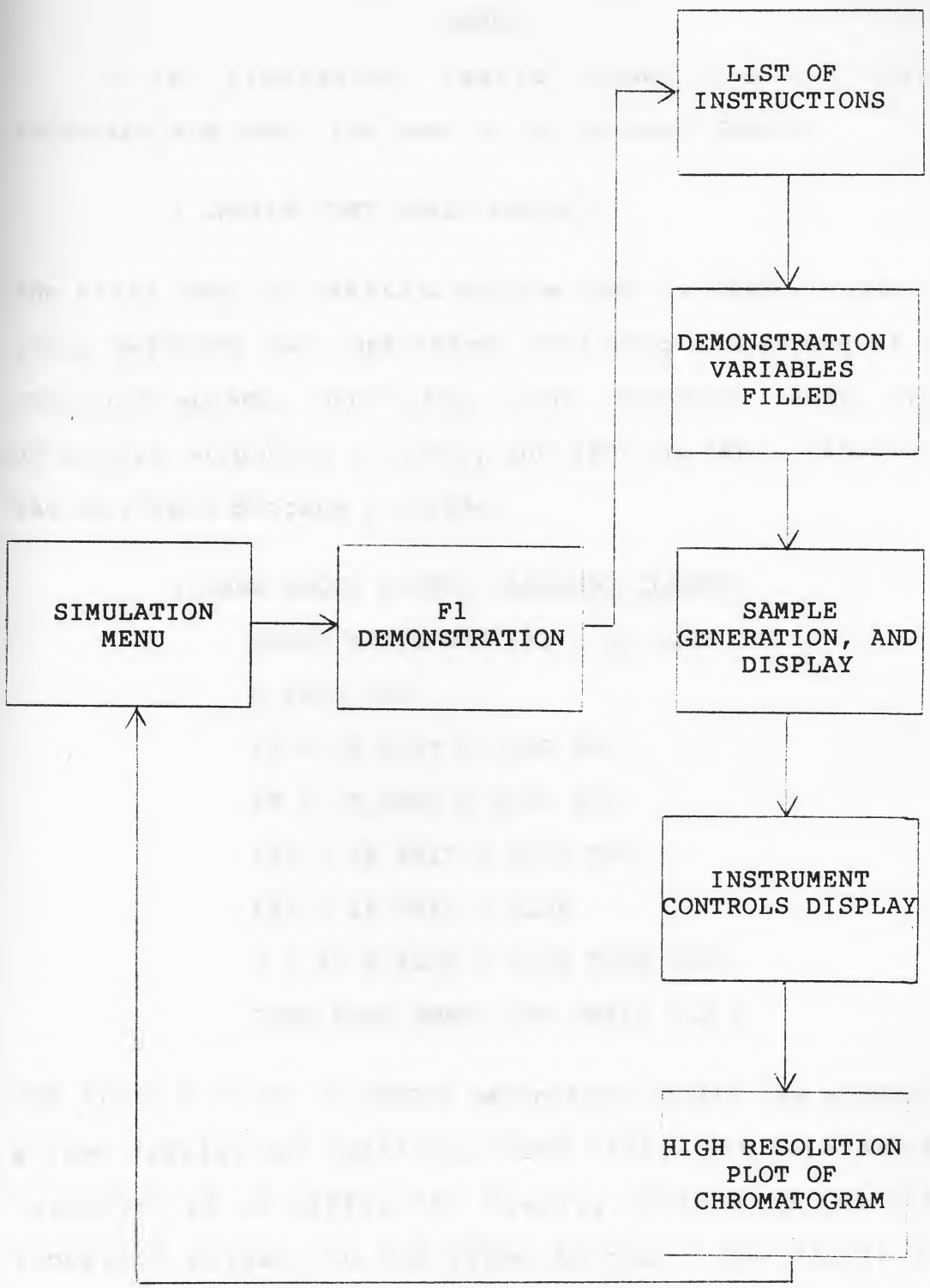

Figure 8. List of operations when DEMONSTRATION is selected on the simulation menu.

SAMPLE

In the simulation, sample composition and column selection are under the control of the word SAMPLE.

```
: SAMPLE CRSR DSPLY PAUSE ;
```

The first word in SAMPLE's source code is CRSR (Cursor), which performs many operations including displaying of the stockroom screen, controling cursor movement, taking input of sample component volumes, and storing those volumes in the DS (Data Storage) buffer.

```
: CRSR STOCK MLSMPL CLMNDSPL CLMNSET
    BEGIN KEYIN DUP 133 = IF DROP CRS
    0 ELSE DUP
    17 = IF EMIT 0 ELSE DUP
    29 = IF EMIT 0 ELSE DUP
    145 = IF EMIT 0 ELSE DUP
    157 = IF EMIT 0 ELSE
    3 = IF 1 ELSE 0 THEN THEN THEN
    THEN THEN THEN LLIM UNTIL CLR ;
```

The first 4 words in CRSR's definition handle the stockroom screen display and updating. STOCK clears the terminal and transfers 1K of buffer RAM memory, containing the blank stockroom screen, to the video display (see figure 9). Next, MLSMPL (Milliliter Sample) updates the screen with zero or with the volume of any component previously entered

GAS CHROMATOGRAPHY

STOCKROOM

ALCOHOLS	ML	ALKANES	ML
METHANOL	00	BUTANE	00
ETHANOL	00	HEXANE	00
PROPANOL	00	OCTANE	00
BUTANOL	00	DECANE	00

AROMATIC	ML	KETONES	ML
BENZENE	00	ACETONE	00
TOLUENE	00	2-BUTANONE	00
O-XYLENE	00	2-PENTANONE	00
P-XYLENE	00	3-PENTANONE	00

COLUMNS: (SET NUMBER 00000)

SQUALANE	O
SE-30	O
DIISODECYL PHTHALATE	O
CARBOWAX 20M	O

PRESS (STOP) KEY TO LEAVE STOCKROOM

Figure 9. Simulator stockroom screen.

into a sample. CLMNDSPL (Column Display) places an X next to the column currently active in the simulation, and CLMNSET (Column SET) diplays the column set number. Screen updating is now complete, and BEGIN marks the start of an

indefinite loop that controls sample data input. Following BEGIN, the source code of CRSR is very similar to MENU's, with program execution being suspended by KEYIN until an entry is made on the keyboard. The ASCII code of the keyboard selection is placed on the stack, and compared to the ASCII codes in CRSR's definition. There are only 6 of C64's ASCII codes in CRSR's definition. They appear in table 3.

ASCII VALUE	C64 KEY
133	F1
17	CRSR DOWN
29	CRSR RIGHT
145	CRSR UP
157	CRSR LEFT
3	RUN/STOP

Table 3. C64 ASCII codes used in CRSR.

The selection of a cursor key will execute the word EMIT. This word transmits the ASCII value on the stack to the selected output device. For example, pressing CRSR UP key will place a 145 on the stack. EMIT interprets the stack value as an ASCII code and moves the cursor one line up. Pressing the RUN/STOP key will set a flag to exit the loop, and pressing f1 executes CRS, initiating the data input and storage routines. Other selections are dropped from the stack, and the program pointers will return to

KEYIN. This type of word programming effectively turns off all but 6 keys on the keyboard, permitting movement of the cursor but limiting the possibilities of screen disruptions.

Execution of CRS sets in motion a rather complex series of data collection words. The words are designed to check all incoming information for accuracy, limiting the possibility of a system crash.

```
: CRS 214 C@ 16 > IF 211 C@ DUP 26 = IF
    DROP 1 ELSE
    66 = IF 1 ELSE 0 THEN THEN ELSE
    211 C@ DUP 13 = IF DROP 1 ELSE DUP
    33 = IF DROP 1 ELSE DUP
    53 = IF DROP 1 ELSE
    73 = IF 1 ELSE 0 THEN THEN THEN
    THEN THEN IF #ENTER THEN ;
```

In line 1 the memory locations 214 and 211 contain the cursor's physical line number, and column number on the screen. The tests in CRS determine the position of the cursor on the screen. They will not permit further execution unless the cursor is under the first column of ML, or over an 0 in the column section. Passing one of these tests executes the word #ENTER.

```
: #ENTER CRSCLR PCLR PAD 1+ 3 EXPECT PAD
    LEN STORE ;
```

This word clears any previous entries, and opens the keyboard for the input of 0 to 3 characters, or until a RETURN is received. The data is temporily stored in PAD, where it is checked by MILCK (Milliliter Check), or CLMNCK (Column Check). If the cursor was in the solute portion of the screen, MILCK determines if the value entered was between 00 and 99. If the cursor is in the column section, CLMNCK checks for the entry of an X. Information other than a number or an X will be rejected.

When a sample entry passes MILCK, the information is passed to the DS (Data Storage) buffer. This buffer is designed to hold all the information about the solutes in the simulator. It is divided into 16 sections called component blocks, each of which contains 58 bytes of data. Figure 10 shows how data are stored in the DS buffer.

The DS memory location for incoming data is selected by ADJ (Adjust).

```
: ADJ DUP 11 > IF 3 - THEN 211 C@ 101
     2 AND 2 = IF 58 ELSE 0 THEN SWAP
     116 * 580 - DS @ + + ! ;
```

The screen column and row numbers are used by ADJ for calculation of the DS memory location. For example, assume an 02 was just entered under the ML column next to ETHANOL. Since ethanol is on the 6th screen line, a 6 will be on the stack from previous checking words. Since the editor mode is

Symbols: n = 16 bit integer

fp = 6 byte floating point number

MEMORY LOCATION	SIZE	CONTENTS
DS 0 +	n	Volume of methanol entered into stockroom screen.
DS 2 +	n	Percentage of total methanol represents.
DS 4 +	n	Buffer memory location for characters spelling METHANOL.
DS 6 +	n	Volume of methanol placed into randomly generated unknown sample.
DS 8 +	n	Blank
DS 10 +	fp	Value of A term in the temperature dependent Antoine equation (eq.15) when Squalane is the selected column.
DS 16 +	fp	B term, Squalane column.
DS 22 +	fp	A term, SE - 30 column.
DS 28 +	fp	B term, SE - 30 column.
DS 34 +	fp	A term, D. Phthalate.
DS 40 +	fp	B term, D. Phthalate.
DS 46 +	fp	A term, Carbowax 20M.
DS 52 +	fp	B term, Carbowax 20M.
DS 58 +	n	Beginning of component block for butane.

Figure 10. Format of first DS buffer Component Block.

is not being used, the C64 looks at the screen in 80 column lines. That is, line 0 contains columns 1 - 40, line 1 contains 41 - 80, and line 2 repeats columns 1 - 40. Therefore the initial two lines in ADJ determine the exact cursor location at the point of input. In the example, ethanol is on an even-numbered line, so the 6 on the stack will not be disturbed. The last line contains the equation that calculates the memory location.

```
( stack value is 6 ) x 116 - 580 = 116
```

The value of 116 is the beginning memory location in the DS buffer for ethanol information. These routines will repeat until the RUN/STOP key is pressed, causing the program to exit the CRSR loop. Volumes stored in the DS buffer by CRSR will then be added together. A percentage for each sample component will be calculated and stored. This is executed by the word DSPLY.

```
: DSPLY % SUM FP@ FO 0 = IF CR
     ." NOTHING IN SAMPLE " CR ELSE
     DISPLAY THEN ;
```

The final sample composition is listed by the word DISPLAY, which loops through the DS buffer checking for solute volumes.

```
: DISPLAY PERCENT SMPLHDR CR 929 0 DO
    DS @ I + @ 0> IF DS @ I 4 + + @
    STKRM @ + 11 TYPE 4 SPACES DS @
    I + ? CR THEN 58 +LOOP ;
```

The name of each solute is displayed by taking the third number stored in a DS buffer component block, adding it to the address in STKRM, and transferring the contents of the next 11 memory locations to the video screen memory. The volumes are then copied from the first memory location in the component block, and displayed.

The final word in SAMPLE is PAUSE, which suspends the program until the operator is ready to continue.

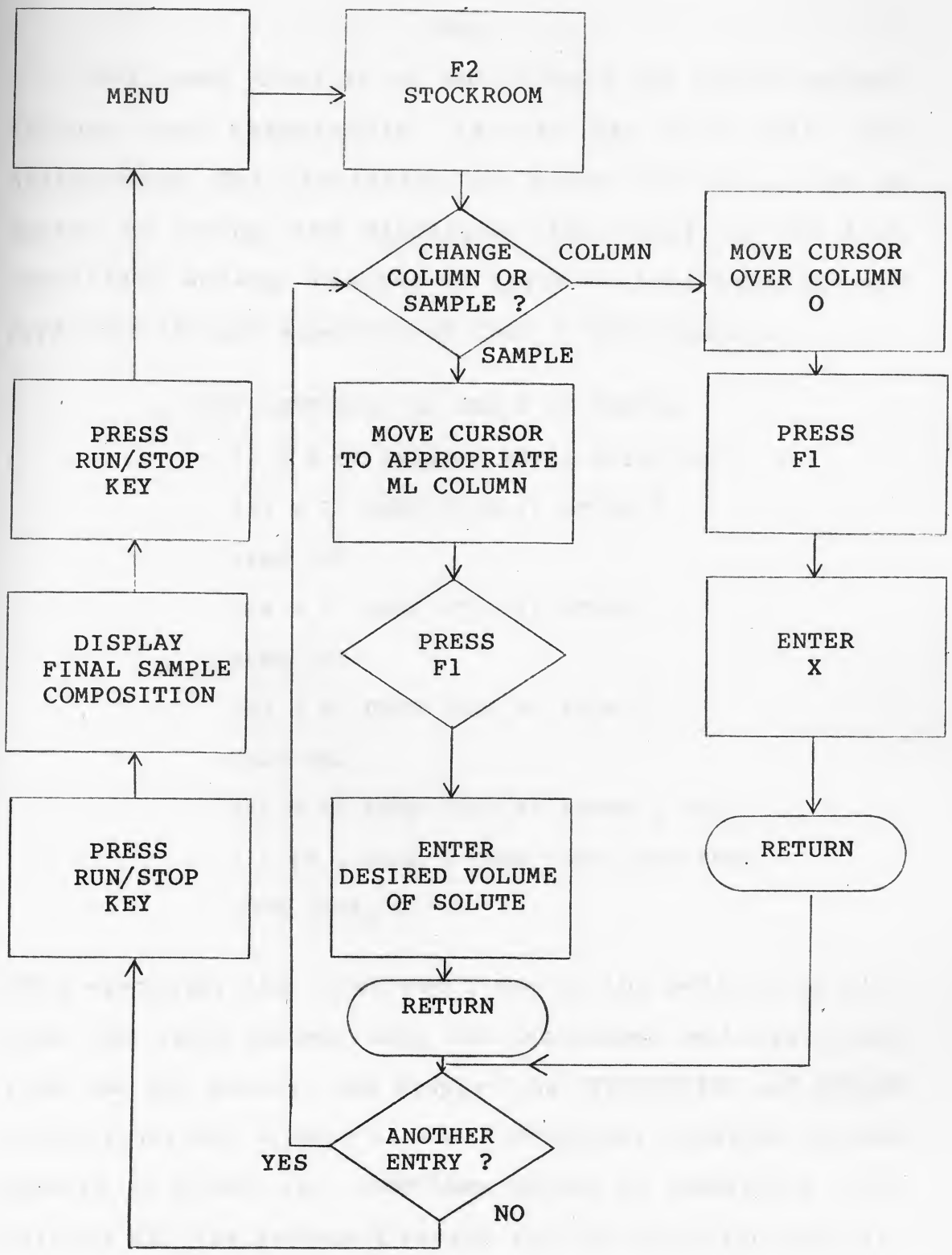

Figure 11. User steps for preparation of a gas chromatograph sample.

INST

Instrument controls on most simple gas chromatographs include oven temperature, carrier gas flow rate, and attenuation. The simulation has these controls, plus an option to change the displayed time range of the high resolution screen. Changes in these control settings are performed through execution of INST (Instrument).

```
: INST INSTDSPLY 30 ATN @ C! ODSPLY
     31 R @ C! FLDSPLY BEGIN KEYIN DUP
     133 = IF DROP FCLR F1 ATTEN 0
     ELSE DUP
     134 = IF DROP FCLR F2 OTEMP 0
     ELSE DUP
     135 = IF DROP FCLR F3 FLOW 0
     ELSE DUP
     136 = IF DROP FCLR F4 RANGE 0 ELSE
     3 = IF 1 ELSE 0 THEN THEN THEN THEN
     THEN UNTIL ;
```

When executed, the first two lines in the definition will clear the video screen, copy the instrument settings screen from the RAM memory, and display the ATTENUATION and SCREEN DISPLAY arrows. A copy of the INSTRUMENT CONTROLS screen appears in figure 12. INST then enters an indefinite loop, turning off the keyboard except for the function keys f1, f3, f5, f7, and the RUN/STOP key. While in the loop, all

settings are accessible with any number of changes permitted.

```
******** INSTRUMENT CONTROLS *********
(F1) ATTENUATOR
      1    2    4    8    16   32   64   128
                     ↑
(F3) OVEN TEMPERATURE   [ 80 TO 200°C ]
     TEMP: _____
(F5) CARRIER GAS
     FLOW RATE  [ 1 TO 125 ML/MIN ]
     FLOW: _____
(F7) DISPLAYED SCANNING RANGE
     0 TO 10 MIN ←
     10 TO 20 MIN
*CURRENT FUNCTION: ____________________

       PRESS (STOP) KEY TO LEAVE
```

Figure 12. Instrument Control screen.

ASCII CODE	C64 KEY
133	F1
134	F3
135	F5
136	F7
3	RUN/STOP

Table 4. ASCII codes used in INST.

On the INSTRUMENT CONTROLS screen, the current attenuation setting is indicated by an upward arrow. When the function key f1 is pressed, the word ATTEN changes the setting by moving the arrow one space to the right.

```
: ATTEN 32 ATN @ C! ATN @ DUP
      1258 = IF DROP 1230 ELSE 4 + THEN
      DUP ATN ! 40 - PCLR PAD 1+ 3 CMOVE
      PAD LEN PAD FPVAL FP 16 FPSWAP FP/
      ATN' FP! 30 ATN @ C! ;
```

The variable ATN holds the current screen memory location of the arrow. By storing a 32 in the memory location, a space is entered onto the screen, thereby erasing the arrow. Movement of the arrow is performed by adding 4 to the value stored in ATN. There is one exception: When 1258 is in ATN, 1230 will be stored in ATN, locating the arrow underneath the first attenuation setting. The value of the new setting is communicated to the program by copying the screen code above the arrow and moving it to PAD. There the length of the string is counted by LEN, and FPVAL converts it to a floating-point number on the stack. The number is divided by 16, and stored in ATN' where it will be used by CURSEN to adjust peak heights. The last step for ATTEN is to place a 30 in the screen memory byte stored in ATN. This operation redisplays the arrow under the appropriate attenuation setting.

Oven temperature is changed through a scroll technique, where the operator presses a CRSR key until the desired temperature is reached. Access to this operation is gained by pressing F3, which executes OTEMP.

```
: OTEMP BEGIN KEYIN DUP
        145 = IF DROP 1 0 +! 0 ORANGE
        ELSE DUP
        17 = IF DROP -1 0 +! 0 ORANGE ELSE
        134 = IF 1 ELSE 0 THEN THEN THEN
        ODSPLY UNTIL O @ 273 + INTFP T FP!
        FCLR ;
```

The loop structure of OTEMP makes scrolling easy. The currently displayed temperature is stored in the variable O. Pressing the up CRSR key will add 1 to this value. Pressing the down CRSR key will subtract 1 from the value in O. ORANGE checks the temperature, to make sure the limits of 80 to 200^{o}C are not exceeded. The temperature is then left as an integer on the stack, where ODSPLY converts it to a string, and stores it as 8-bit bytes in the screen video memory locations next to TEMP.

```
: ODSPLY OCLR P$CLR O @ STR$ 1+ 1396
        5 CMOVE ;
```

Exiting the temperature loop is performed by pressing f3, which sets a flag for UNTIL. The last temperature entered

into O is then fetched, converted to Kelvin by adding 273, and the result is stored in the variable T.

Changes in carrier gas flow rate are performed by the word FLOW, which has a source code similar to OTEMP. The only differences are range limits, and variables where the final selection is stored.

The instrument control screen contains a selection called DISPLAYED SCANNING RANGE. This option tells the chromatogram plotting words for what time frame results should be displayed. Since only 2 time ranges are offered, the controlling word RANGE is rather simple.

```
: RANGE R @ 1764 = IF 31 1804 DUP R ! C!
     640 RNG ! ELSE
     31 1764 DUP R ! C! 0 RNG ! THEN ;
```

The variable R holds the screen memory location for the arrow. This value is fetched to determine the arrow location, and then switched to the other position. Depending on the setting a 0 or 640 will be stored in RNG. This variable is used by the words PLT1 and PLT2 to set scanning ranges

The last option in INST is the RUN/STOP key, which sets the flag to exit the loop, and returns control to the menu.

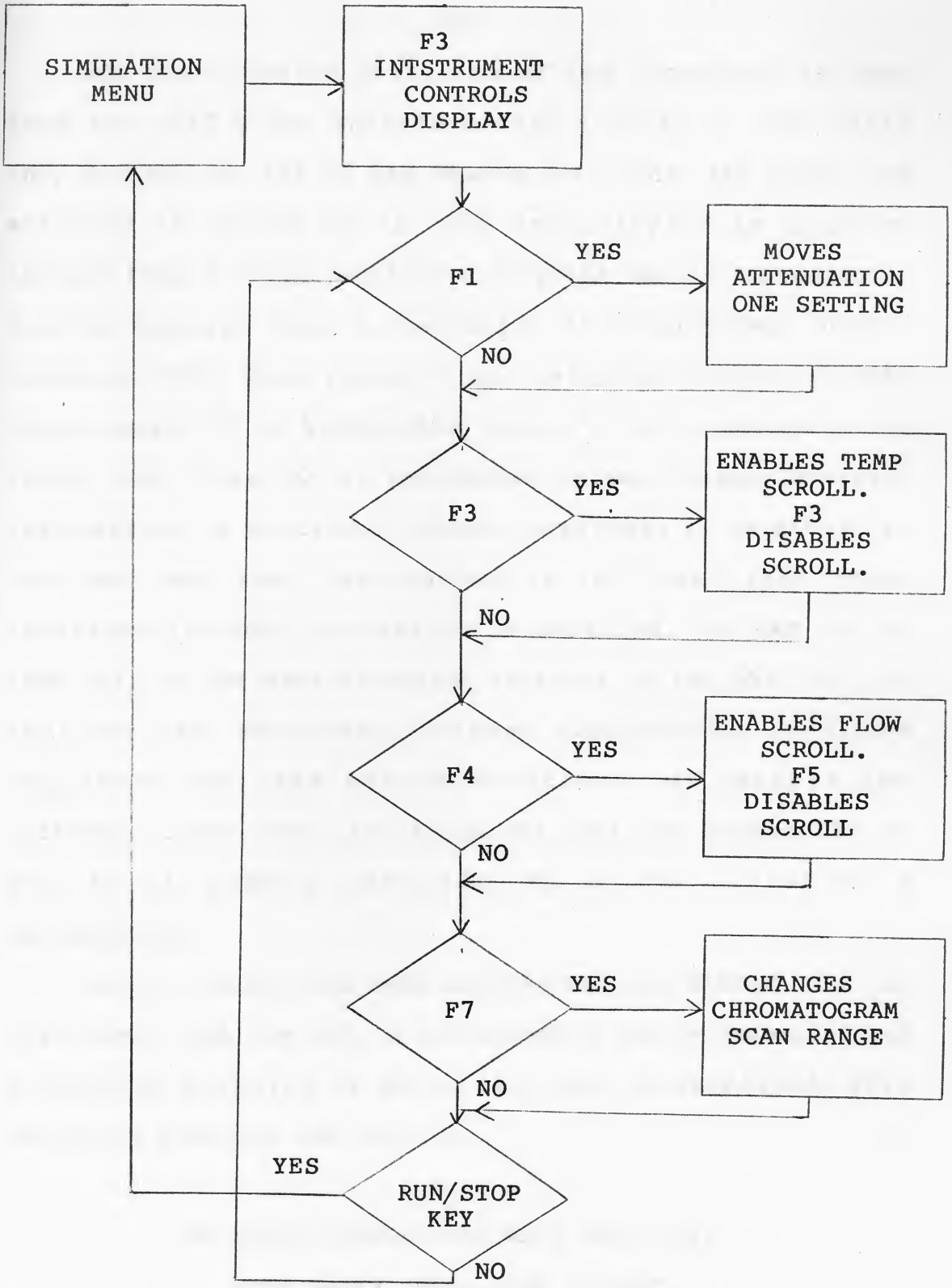

Figure 13. User options on instrument control screen.

GC

All the graphics abilities of the Commodore 64 come from the 6567 Video Interface Chip (VICII). The VICII chip divides the 64K of RAM memory into four, 16K banks. The standard 40 column by 25 line text display is accessed through bank 0. High resolution graphics can be conveniently handled through bank 3, beginning at hexadecimal memory location C000. This location was selected because at E000 there exists 8K of hidden RAM memory lying underneath the kernal ROM. This RAM is considered hidden because whenever information is written to these locations, it is stored in the RAM. But when information is retrieved from these locations the ROM information is recalled. The RAM can be read only if the data direction register on the 6510 chip is switched off. This means through manipulation of 6510's registers the same memory locations can perform two different tasks. The simulation utilizes the hidden RAM as the 8K of memory necessary to store a plot of a chromatogram.

After a sample has been entered through SAMPLE, and the instrument controls set, a chromatogram can be generated and plotted by pressing f4 while the menu is displayed. This selection executes the word GC.

```
: GC DELAY PEAKS HIRES AXIS HDRS PLT1
     PLT2 MARK2 CROSS 0BNK 1 SPOFF ;
```

Within GC's source code the word DELAY executes HIRES, which switches the text display to the 320 by 200 pixel high resolution display.

```
HEX ( DEFINITION IS IN HEXADECIMAL )
: HIRES 3BNK SETBM CLRBM 10 FILSCRN
    0 CURX ! 0 CURY ! ;
```

3BNK sets the registers on the VICII chip to the 3rd bank, and SETBM switches the chip into the bit-map mode. The 8K of RAM memory starting at E000 is filled with zeros, by CLRBMM. When in the bit-map, mode the locations designated as screen memory will be used for color information. The upper 4 bits of each screen memory byte determine the color of any bit that is set to 1, and the lower 4 bits determine the color of any bit set to a 0. Therefore by placing a hexadecimal 10 before the word FILSCRN, a screen of white and black pixels is created. The variables CURX and CURY designate the pixel where any drawing routine will begin.

Since the VICII chip is set to the 3rd bank, the normal character printing routines are not operational. Therefore when DELAY prints out the enlarged letters GAS CHROMATOGRAPH, it uses the word RCHR. This word will recall the bit pattern of any ROM character, by designating the character code. RCHR also requires the location on the high resolution screen where the character is to be printed. For example :

```
8 8 192 RCHR
```

will place a letter H beginning at pixel (8, 192) which is in the lower left corner of the screen. The size of the character is controlled by ZOOM. Storing a 4 in this variable will create a character 4 times the normal size.

Following DELAY, GC executes PEAKS, which combines RETTIME, SIGMA, and AREA to calculate peak locations and shapes. The resultant chromatograms are stored in 2 buffers identified as PB and PB2. HIRES is executed again to clear the screen, and AXIS divides the screen to display two portions of the chromatogram. Lines are then drawn by the word LINES.

```
: LINES 0 88 319 88 LINE
        0 176 319 176 LINE ;
```

The first two numbers in the definition designate the pixel where a line is to begin. The last two numbers designate where the line ends.

Characters spelling TR, MIN, and HEIGHT are placed in the lower left corner by HDRS (Headers). This word, like DELAY, uses RCHR to call up the designated ROM characters. ZOOM is set to 0, so characters are normal size.

After the formatting and labeling of the screen comes the plotting of the chromatogram. The upper portion of the screen uses PLT1 to transfer the data from the PB buffer, while the lower half uses PLT2.

```
: PLT1 0 CURX ! 85 CURY !
        320 0 DO I PB @ I 2* + @
        85 SWAP - LINETO LOOP ;
```

The chromatogram data are stored sequentially in the PB buffer, so PLT1 uses the the loop index I as the pixel x coordinate, and recalls the y coordinate from the buffer. Since the top of the screen has a line number of 0, each y coordinate must be subtracted from 85 to prevent the chromatogram from being drawn upside down. LINETO connects the pixel in CURX and CURY to the pixel placed on the stack by the loop . It then updates CURX and CURY with the stack values. All drawing is performed by machine language routines accessed by FORTH words, so litle time is lost on plotting chromatograms. For example, an entire chromatogram which consists of connecting 640 pixels, takes about 2.5 seconds to plot. The plotting time will vary slightly depending on the number, and height of the peaks in the chromatogram, but this variance does not exceed .5 seconds. This is very quick when compared to bit mapping routines written directly in FORTH. A chromatogram consisting of connecting only 320 pixels, plotted by FORTH words averages about 23 seconds to complete.

When a peak is being formed, variables hold its exact retention time and standard deviation values. Most quantitative measurements are taken at a peak width of 4σ. Therefore to increase the accuracy of peak width

measurements, MARK2 places two small lines at the base of each peak signifying the location of 4σ.

An additional feature of the simulation is the movable cross, which aids in determining retention times and peak areas. This is accomplished by moving the cross to the peak maximum and pressing F1. The peak height and retention time will be displayed.

The cross is actually a SPRITE that is maintained directly by the VICII chip. The SPRITE can float over the entire screen without disturbing the chromatogram underneath. CROSS is the controlling word for SPRITE movement, and position updating.

```
: CROSS 23 SPX ! 49 SPY ! TR# HGHT SPRT
    BEGIN KEYIN DUP 3 = IF DROP 1 ELSE DUP
    17 = IF 1 SPX @ SPY @ 1+ DUP LIMY
    SPY ! SPXY DROP 0 ELSE DUP
    29 = IF 1 SPX @ 1+ DUP LIMX SPX !
    SPY @ SPXY DROP 0 ELSE DUP
    145 = IF 1 SPX @ SPY @ 1- DUP LIMY
    SPY ! SPXY DROP 0 ELSE DUP
    157 = IF 1 SPX @ 1- DUP LIMX SPX !
    SPY @ SPXY DROP 0 ELSE DUP
    133 = IF TR# HGHT DROP 0 ELSE DUP
    CRMOV DROP 0 THEN THEN THEN THEN
    THEN THEN UNTIL ;
```

The variables SPX and SPY contain the current location of the cross center. The words TR# and HGHT take the SPRITE location, convert it to ROM number characters and display it in the lower corner, next to TR and HEIGHT. One option for cross movement is to press one of the CRSR keys. From the definition it can be seen that this will add or subtract one pixel in the direction designated on the key. A SPRITE can travel outside the boundaries of the visible viewing area, so LIMX and LIMY restrict cross movement to within the chromatogram format. An option also exists to speed up the cross movement. This is controlled by the word CRMOV which is programmed in the same manner as CROSS, except that it uses the function keys f5 - f8, and each entry will move the SPRITE 8 pixels. Pressing f1 will update the cross position by executing TR# and HGHT.

The RUN/STOP key exits the cross loop and 0BNK switches the VICII chip back to 0 bank, and standard text display. SPOFF turns the cross off, and program control goes back to the menu.

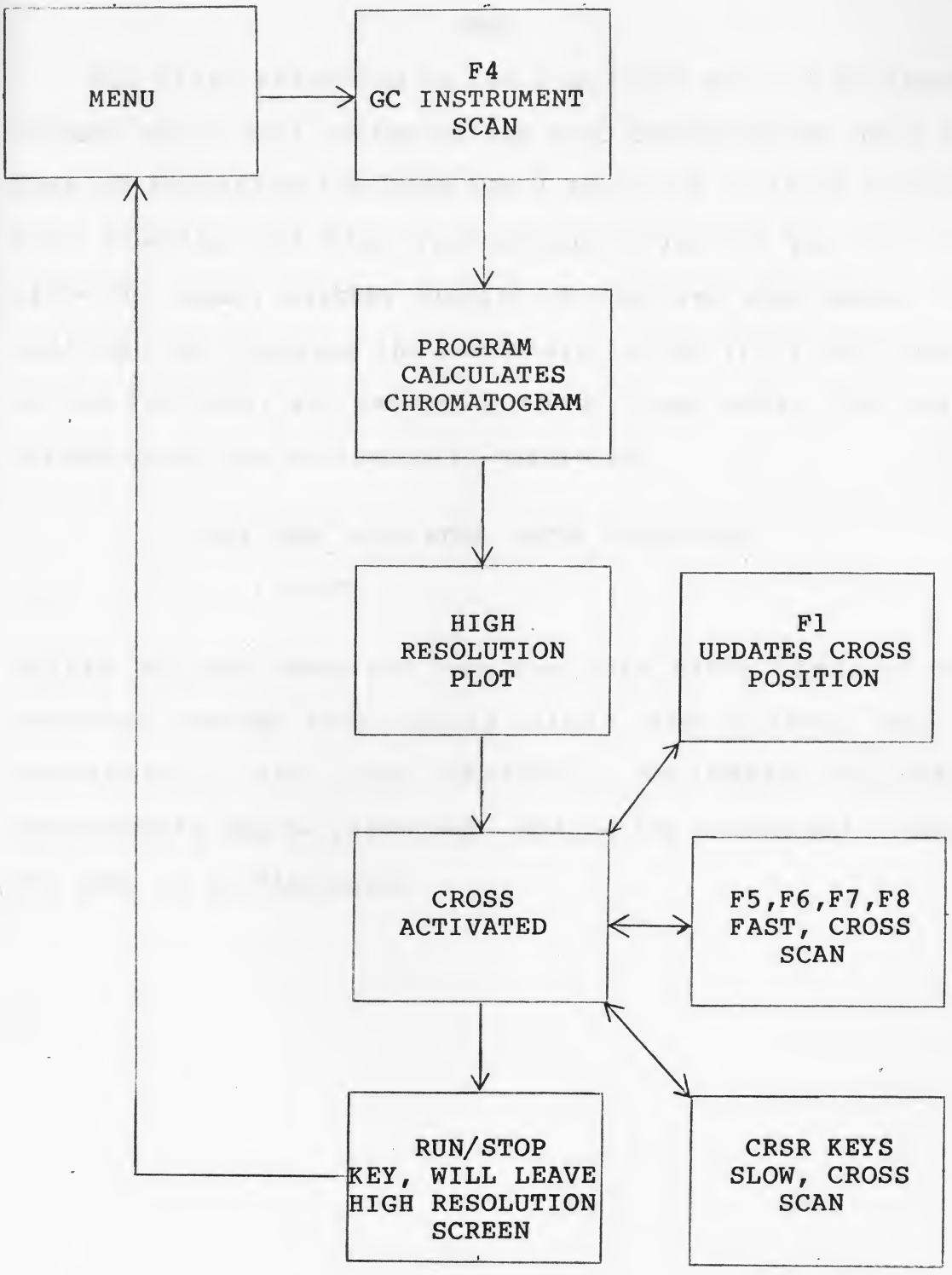

Figure 14. Operations, and function keys controlled by the word GC.

PRV

The fifth selection on the simulator menu is PREVIOUS SCREEN, which will redisplay the last chromatogram. This is done by executing the word PRV (Previous). Since normal text display and high resolution graphics are in two different banks, neither display is modified when banks are switched. By changing the registers on the VICII chip back to the 3rd bank, and resetting the bit-map mode, the last chromatogram can be instantly displayed.

```
: PRV 3BNK AXIS HDRS SETBM CROSS 0BNK
      1 SPOFF ;
```

Unlike GC, PRV does not use the word HIRES, because it contains words that would clear the screen. As a convenience, the cross SPRITE is activated, so peak measurements can be performed. Exiting the screen will cause the menu to be displayed.

HETP

The sixth selection on the menue is labeled " HETP EXPERIMENT ", which displays 2 screens that explain how to conduct an HETP experiment. The text is copied from the buffer RAM, and transfered to the video output through the word TYPE.

```
: HETP CLR B2 @ 880 TYPE PAUSE
       CLR B1 @ 360 TYPE DSCLR PAUSE ;
```

The variables B1 and B2 contain the beginning addresses of the buffers that contain the text. Numbers prior to the word TYPE tell how many characters will be transferred. DSCLR clears any previous samples, so the experiment can begin with a single solute. At the end of each line, PAUSE is used, so the operator has control over how long each screen is displayed.

UNKNOWN

The simulator has an option for an operator to analyze a sample containing two unidentified components. This sample is generated by the word RNDM.

```
: RNDM DSCLR RNDM2 RNDM4 ROT DS @ + 6 +!
    SWAP DS @ + 6 +! ;
```

Execution of RNDM will clear any previous samples entered through the STOCKROOM screen. Then, under the control of RNDM2 and RNDM4, a series of 4 random number generators begin to work. RNDM2 has the job of selecting the sample components, and RNDM4 generates the component volumes. To explain how this is performed, it is useful to recall the format of the DS buffer. Within the DS buffer there are 16 component blocks, each containing 58 bytes of memory. These blocks are divided into sections that contain information on each solute. Using the real-time jiffy clock as a seed value, RNDM2 is programmed to select 2 numbers from a field of 16. The numbers available are in increments of 58, that is 0, 58, 116, --- 870. Each of these numbers represents the beginning of a component block. A second set of random number generators in RNDM4 picks 2 distinct numbers from a field of 1 thru 7. The 4 numbers selected are identified as an unknown sample number. Like the column set number, the unknown number is a code that tells the contents of the sample. For example, a code number could be:

The number is decoded by dividing 116 by 58 (the length of a component block) and adding 1, for a result of 3. Going from left to right on the stockroom screen, the 3rd chemical is ethanol. Applying the same formula to 522 will result in a 10, making the second component acetone. The last 2 numbers represent the component volumes, meaning 3 milliliters of ethanol, and 5 milliliters of acetone are in the sample. Whenever an unknown number is entered into the program only the volumes are stored in the DS buffer. This permits use of all simulator operations without disclosure of the sample contents.

The controlling word for the unknown experiment is UNKNOWN.

```
: UNKNOWN CLR DSCLR CR CR BEGIN MSG6
        KEYIN DUP 89 = IF DROP ?IN ?DECODE
        % PERCENT 1 ELSE
        78 = IF 1 CLR CR CR MSG8 RNDM %
        PERCENT PAUSE ELSE 0
        REDO THEN THEN UNTIL ;
```

The first 3 lines of UNKNOWN deal with entering a previously generated sample number. If an operator has a sample number, ?IN will accept it, and ?DECODE will place the volumes in the DS buffer. If there is no number, RNDM will generate one. If the operator's input is incorrect, REDO will ask him to repeat the entry.

PCMPR

Once a gas chromatographic separation has been performed on the unknown sample, and components identified, there is an option that checks the sample composition. This option is the 8th selection on the simulator menu, and it executes the word PCMPR.

```
: PCMPR DSCLR ?IN ?DECODE % PERCENT
        BEGIN PCOMPARE CR BEGIN MSG10 KEYIN
        CLR CR CR DUP
        89 = IF DROP 0 1 ELSE
        78 = IF 1 1 ELSE REDO 0 THEN THEN
        UNTIL UNTIL ;
```

Answer checking begins with entering the unknown number through ?IN. The number is temporarily stored in PAD, where ?DECODE breaks it apart into component blocks and stores component volumes in the DS buffer. A rather complex word called PCOMPARE will then perform the actual answer checking.

```
: PCOMPARE CR MSG9 PCLR INPUT PAD LEN
    986 0 DO DS @ I + 6 + @ 0> IF
    DS @ I + 4 + @ STKRM @ + PAD 30 +
    12 CMOVE PADJ PMOVE PEQUIL IF MSG1
    MSG2 ?NBRCK DS @ I + 2+ @ ?% IF
    MSG3 LEAVE ELSE MSG4 LEAVE THEN
    ELSE PCLR2 THEN THEN I 928 = IF
    MSG5 THEN 58 +LOOP ;
```

PCOMPARE's first move is to ask for the name of one component. The entry is placed in PAD, where each character is compared to the names stored in the STKRM buffer. All the comparisons take place in the PAD memory. If a component within unknown sample is correctly identified, the program will ask how much solute is present. The operator will enter a percentage based on peak area. This entry is checked against the percentage calculated by PERCENT and stored in the DS buffer.

Since PCMPR is programmed as an indefinite loop the operator has an unlimited number of tries at getting the right answer.

SAMPLCLR & NEW

The last two options on the simulator menu are clearing words. When an operator wants to dispose of a sample, a selection of CTRL 1 from the menu will accomplish the task. SMPLCLR is the controlling word, and its execution clears the volumes, percentages, and unknown selections in the DS buffer. The word gives a 1 line statement saying the sample has been cleared.

```
: SAMPLCLR CLR CR CR CR B2 @ 880 +
        40 TYPE CR CR DSCLR PAUSE ;
```

The simulation program is a turnkey program, so the last menu selection, " LEAVE THE SIMULATOR ", is somewhat deceiving. The program cannot be exited unless the computer is turned off. But pressing CTRL 2 will execute NEW, which leaves the menu loop. The word NEW will also clear samples, chromatograms, and column set numbers, used in previous work.

```
: NEW DSCLR PBCLR STARTUP COLUMNS ;
```

V. EXPERIMENTS and RESULTS

HETP experiments were performed with 3 different solutes using various columns available in the simulation. A new column set number was generated prior to each separation. The symbols used in the data tables are:

FLOW - Carrier gas flow rate in ml/min.

μ - Linear gas velocity in cm/second.

AIR - Retention time of air in min.

x - Distance from injection to peak maximum.

y - Length of peak baseline cut by two tangents.

N - Number of theoretical plates.

H - Height equivalent to a theoretical plate in cm.

Column set number: 44150

Column: SE-30

Temperature: 110°C

Attenuation: 16

Sample: Hexane

FLOW	μ	AIR	x	y	N	H
15	1.18	1.73	8.44	1.4	581	.210
20	1.56	1.30	6.33	.95	710	.172
30	2.36	.86	4.23	.61	769	.159
40	3.18	.64	3.16	.44	825	.148
50	3.99	.51	2.53	.36	790	.154
60	4.73	.43	2.10	.30	784	.156
70	5.50	.37	1.81	.27	719	.170
80	6.35	.32	1.56	.25	623	.196
90	7.26	.28	1.39	.24	552	.221
100	7.82	.26	1.25	.21	567	.215
120	9.68	.21	1.05	.19	488	.250

Table 5. HETP experiment using hexane on a SE-30 column.

Column Set Number: 44150

Column: SE-30

Temperature: 100°C

Solute: Hexane

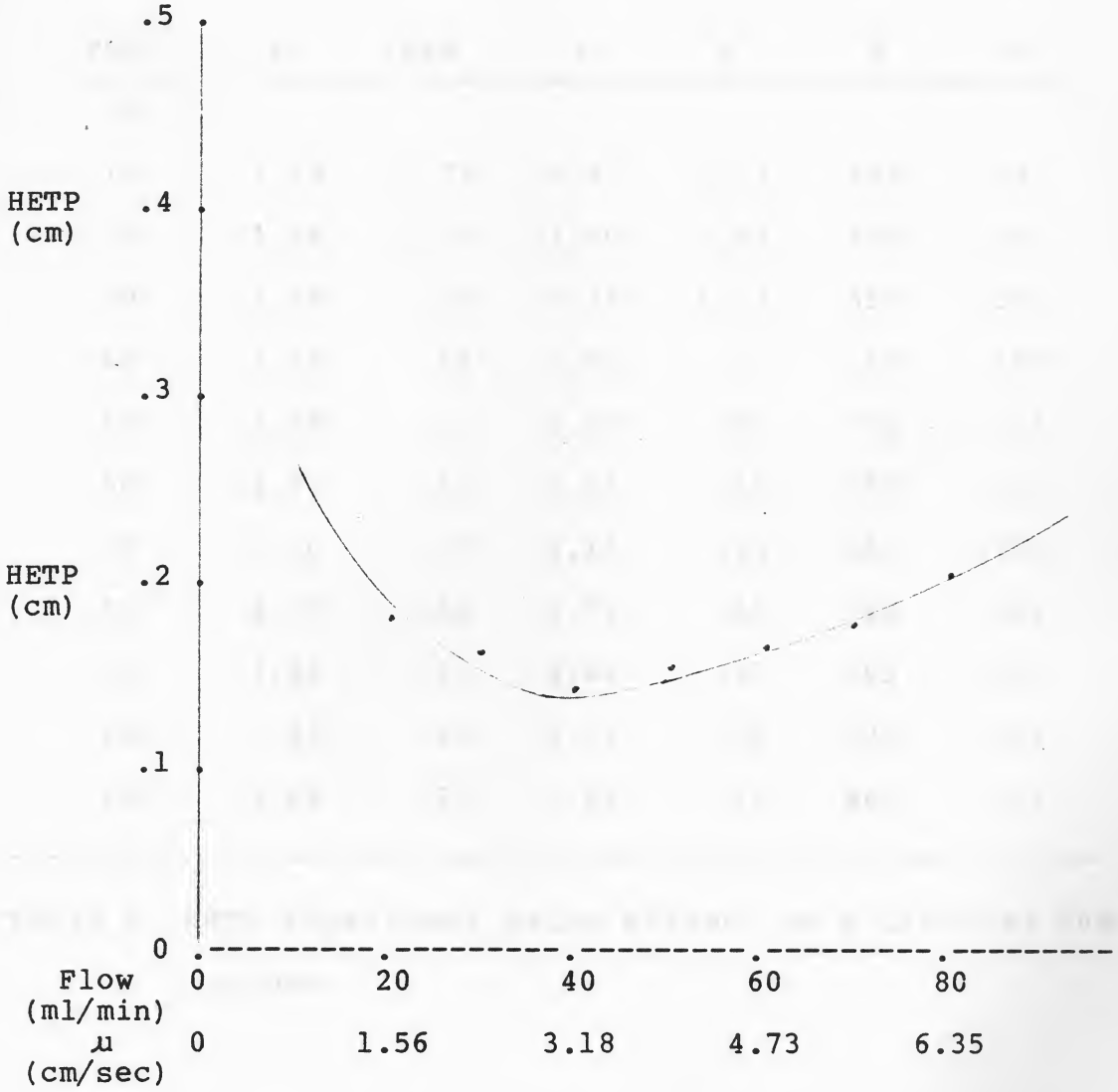

Figure 15. HETP versus linear gas velocity using hexane on a SE-30 column.

Column set number: 39257

Column: Carbowax 20M

Temperature: 110°C

Attenuation: 16

Solute: Ethanol

FLOW	μ	AIR	x	y	N	H
10						
15	1.18	1.73	14.67	2.33	634	.192
20	1.56	1.30	11.00	1.64	720	.169
30	2.36	.86	7.33	1.03	810	.151
40	3.18	.64	5.50	.77	816	.150
50	3.99	.51	4.40	.64	756	.161
60	4.73	.43	3.66	.54	735	.166
70	5.50	.37	3.15	.49	661	.185
80	6.35	.32	2.75	.45	598	.204
90	7.82	.26	2.44	.42	565	.225
100	7.82	.26	2.19	.39	504	.242
120	9.68	.21	1.83	.34	463	.264

Table 6. HETP experiment using ethanol on a Carbowax 20M column.

Column Set Number: 39257

Column: Carbowax 20M

Temperature: 110°C

Solute: Ethanol

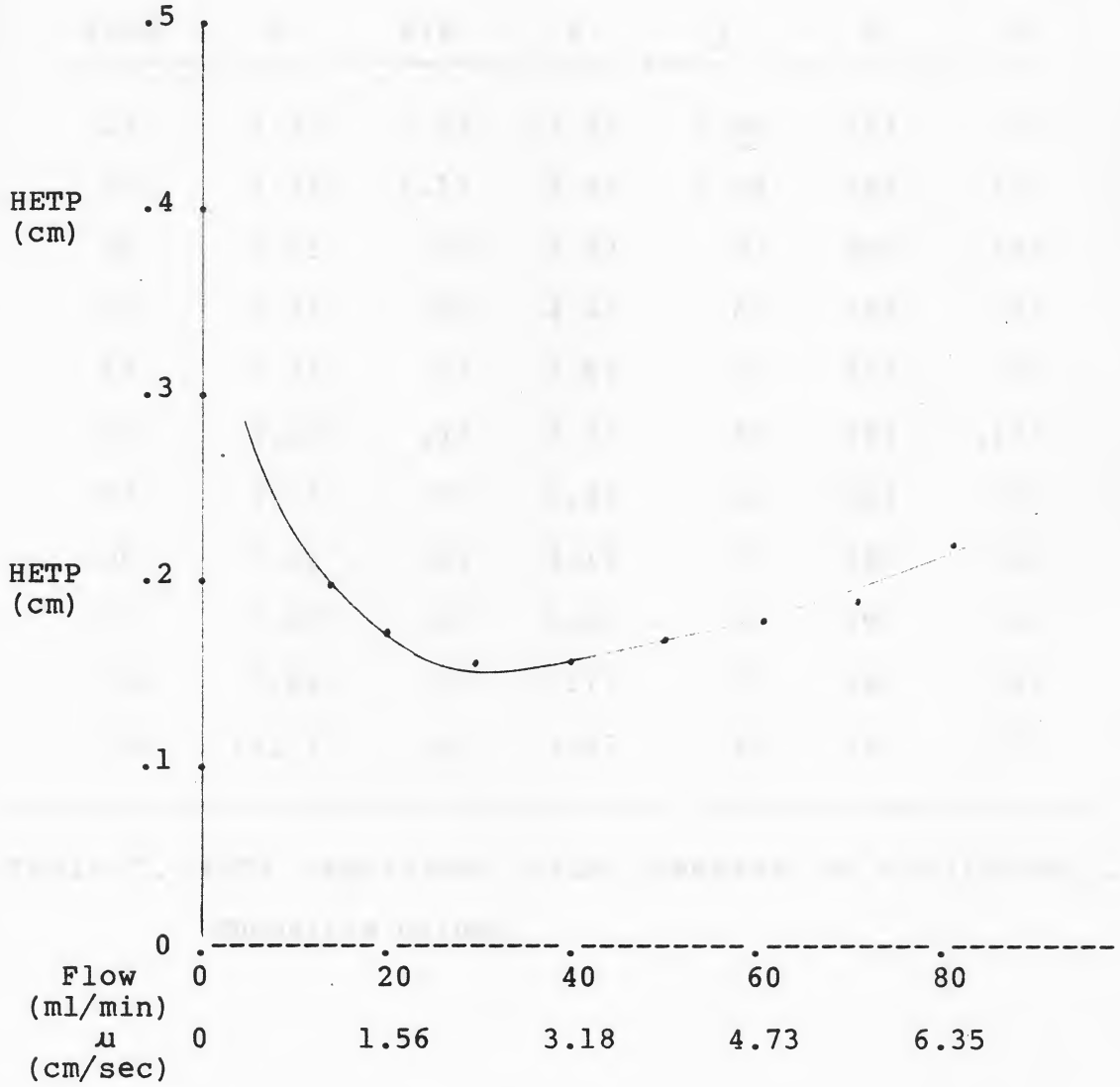

Figure 16. HETP versus linear gas velocity using ethanol on a Carbowax 20M column.

Column set number: 39457

Column: Diisodecyl Phthalate

Temperature: 150°C

Attenuation: 16

Sample: Benzene

FLOW	μ	AIR	x	y	N	H
15	1.30	1.56	11.75	1.84	653	.187
20	1.74	1.17	8.82	1.29	748	.163
30	2.61	.78	5.87	.83	800	.153
40	3.45	.59	4.42	.63	788	.155
50	4.33	.47	3.52	.52	733	.166
60	5.10	.39	2.94	.45	683	.179
70	6.35	.32	2.52	.41	604	.202
80	7.01	.29	2.19	.36	592	.206
90	7.82	.26	1.95	.35	497	.245
100	8.84	.23	1.77	.33	460	.265
120	10.17	.20	1.47	.28	441	.277

Table 7. HETP experiment using benzene on a Diisodecyl Phthalate column.

Column Set Number: 39457

Column: Diisodecyl Phthalate

Temperature: 150°C

Solute: Benzene

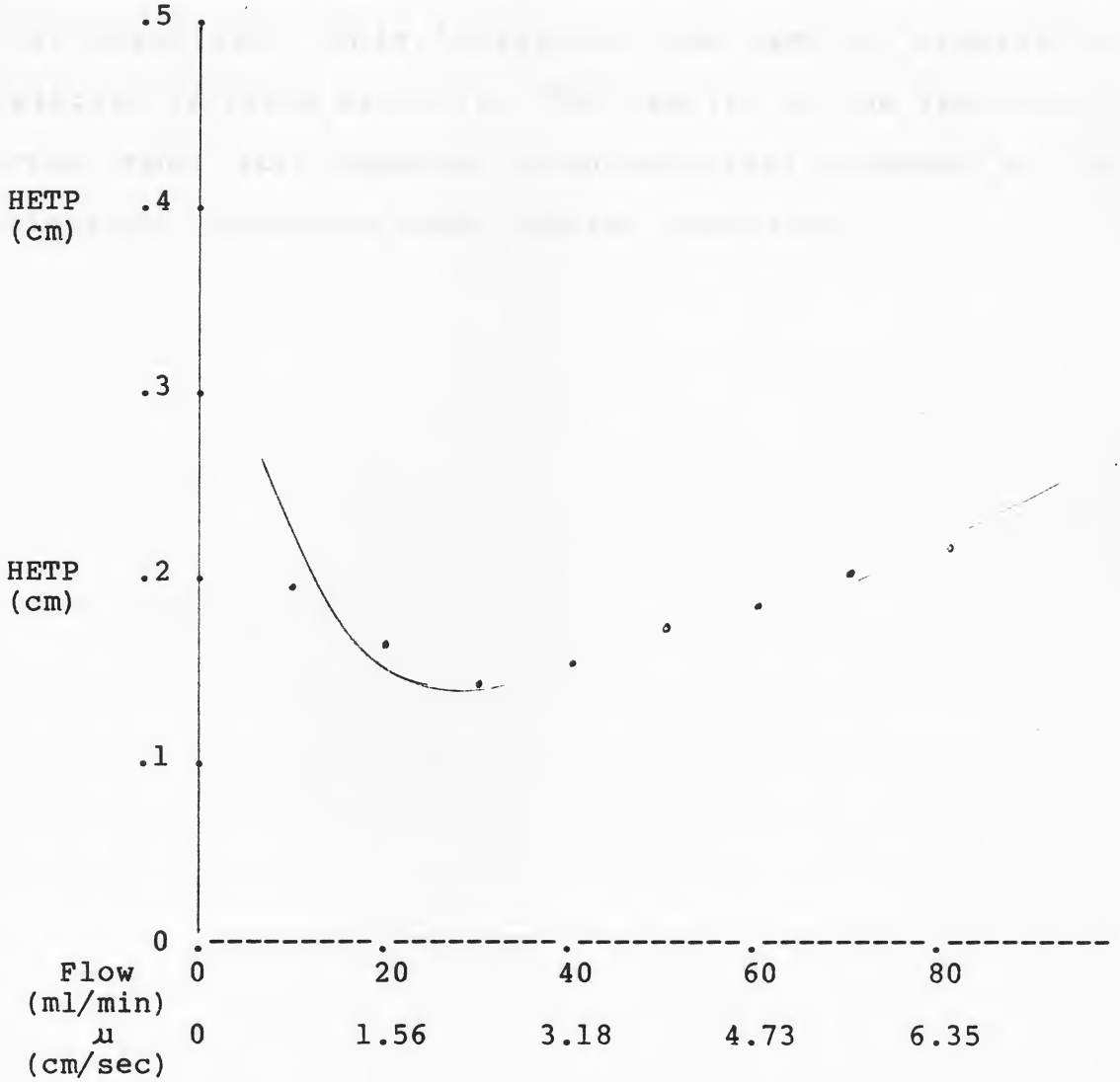

Figure 17. HETP versus linear gas velocity using benzene on a Diisodecyl Phthalate column.

The simulator was developed to generate chromatograms similar to the ones obtained with laboratory instruments. To determine if this was achieved, students in a chemistry 401/601 chromatography course, experimented with a number of solutes on GOW-MAC gas chromatographs. Additionally, one student had access to a Perkin-Elmer Sigma 300 gas chromatograph. This instrument was used to separate a mixture of three alcohols. The results of the laboratory experiments were compared to chromatograms generated by the simulator, operating under similar conditions.

LABORATORY DATA

Gas Chromatograph: Gow Mac
Column: SE-30 (DC-200)
Flow Rate: 50 ml/min
Temperature: 100°C
135°C

Solute	AIR (min)	t_R (min)	t'_R (min)	V'_R (ml)	$\frac{V'_R}{V'_R(\text{hexane})}$
Hexane	.37	.98	.61	30.5	1.0
	.33	.53	.20	10.0	1.0
Acetone	.42	.70	.28	14.0	.46
	.34	.51	.17	8.5	.85
2-Butanone	.41	1.03	.62	31.0	1.02
	.35	.60	.25	12.50	1.25
2-Pentanone	.38	1.34	.96	48.0	1.57
	.35	.75	.40	20.0	2.0
3-pentanone	.40	1.50	1.10	55.0	1.80
	.36	.79	.43	21.50	2.15

Table 8. GOW-MAC retention data using a SE-30(DC-200) column.

LABORATORY DATA

Gas Chromatograph: Gow Mac
Column: Carbowax 20M
Flow Rate: 50 ml/min
Temperature: 100°C
135°C

Solute	AIR (min)	t_R (min)	t'_R (min)	V'_R (ml)	$\frac{V'_R}{V'_R(\text{hexane})}$
Hexane	.43	.57	.14	7.0	1.0
	.33	.38	.05	2.5	1.0
Toluene	.45	2.18	1.73	86.5	12.36
	.33	.78	.45	22.5	9.0
Acetone	.41	.82	.41	20.5	2.93
	.34	.52	.18	9.0	3.60
2-Butanone	.45	1.22	.77	38.5	5.50
	.33	.57	.24	12.0	4.80
2-Pentanone	.45	1.55	1.10	55.0	7.86
	.33	.64	.31	15.5	6.2
3-Pentanone	.38	1.54	1.16	58.0	8.29
	.33	.63	.30	15.0	6.0

Table 9. GOW-MAC chromatograph retention data using a Carbowax 20M column.

LABORATORY DATA

Gas Chromatograph: Perkin-Elmer Sigma 300
Column: Carbowax 20M
Flow Rate: 20 ml/min
Temperature: 100°C
135°C

Solute	AIR (min)	t_R (min)	t'_R (min)	V'_R (ml)	$\frac{V'_R}{V'_R(\text{hexane})}$
Hexane	.52	.71	.19	3.80	1.0
	.49	.62	.13	2.60	1.0
Methanol	.52	1.56	1.04	20.8	5.47
	.49	.95	.46	9.20	3.54
Ethanol	.52	1.79	1.27	25.4	6.68
	.49	1.02	.53	10.6	4.08
Propanol	.52	2.95	2.43	48.6	12.79
	.49	1.44	.95	19.0	7.31

Table 10. P-E Sigma 300 retention data using a Carbowax 20M column.

SIMULATION DATA

Column Set Number: 43451
Column: SE-30
Flow Rate: 50 ml/min
Temperature: 100°C
135°C

Solute	AIR (min)	t_R (min)	t'_R (min)	V'_R (ml)	$\frac{V'_R}{V'_R(\text{hexane})}$
Hexane	.51	3.05	2.54	127.0	1.0
	.47	1.67	1.20	60.0	1.0
Methanol	.51	1.14	.63	31.5	.25
	.47	.79	.32	16.0	.27
Ethanol	.51	1.55	1.04	52.0	.41
	.47	.95	.48	24.0	.19
Propanol	.51	2.64	2.13	106.5	.84
	.47	1.42	.95	47.5	.79
Toluene	.51	9.02	8.51	425.5	3.35
	.47	3.94	3.47	173.5	2.89
Acetone	.51	1.66	1.15	57.5	.45
	.47	1.06	.59	29.5	.49
2-Butanone	.51	2.97	2.46	123.0	.97
	.47	1.58	1.11	55.5	.93
2-Pentanone	.51	5.07	4.56	228.0	1.80
	.47	2.42	1.95	97.5	1.63
3-Pentanone	.51	5.34	4.83	241.5	1.90
	.47	2.50	2.03	101.5	1.69

Table 11. Simulation retention data using a Carbowax 20M column.

SIMULATION DATA

Column Set Number: 43451
Column: Carbowax 20M
Flow Rate: 50 ml/min
Temperature: 100°C
135°C

Solute	AIR (min)	t_R (min)	t'_R (min)	V'_R (ml)	$\frac{V'_R}{V'_R(\text{hexane})}$
Hexane	.51	1.34	.83	41.5	1.0
	.47	.89	.42	21.0	1.0
Methanol	.51	4.79	4.28	214.0	5.16
	.47	2.16	1.69	84.5	3.93
Ethanol	.51	5.76	5.25	262.5	6.33
	.47	2.44	1.97	98.5	4.58
Propanol	.51	10.57	10.06	503.0	12.12
	.47	3.98	3.51	175.5	8.36
Toluene	.51	13.58	13.17	685.5	15.87
	.47	5.47	5.0	250.0	11.90
Acetone	.51	3.52	3.01	150.5	3.63
	.47	1.84	1.37	68.5	3.26
2-Butanone	.51	5.73	5.22	261.0	6.29
	.47	2.66	2.19	109.5	5.21
2-Pentanone	.51	9.09	8.58	429.0	10.34
	.47	3.80	3.33	166.5	7.93
3-pentanone	.51	9.09	8.58	429.0	10.34
	.47	3.76	3.29	164.5	7.83

Table 12. Simulator retention data using a SE-30 column.

The shape of the curves in the HETP versus linear gas velocity graphs, indicate some important points about the simulator's ability to emulate real world chromatography behavior. The variations of H with changing carrier gas flow rates are representative of the behavior predicted by the van Deemter equation. This means the equations used to determine peak shape accurately duplicate the contributions predicted for various peak-broadening mechanisms. The optimum flow rates predicted by the graphs, show that the random number generators do create columns of differing physical characteristics. Therefore in the areas of peak shape, and column performance, the simulation does a good job in predicting laboratory instrument behavior.

Duplication of retention behavior between simulator and lab instrument, is determined by comparing the ratios of V'_R/V'_R(hexane). The adjusted retention volume ratios for the alcohols separated on the Carbowax 20M/k-40M column using the Sigma 300 chromatograph, correlated very well with the simulator's adjusted retention volumes, using the Carbowax 20M column. Correlation between simulator data, and data obtained on the GOW-MAC chromatographs decreased somewhat, but still showed similar behavior patterns. These results plus the HETP curves, show that the simulator has succeeded in combining GLC theory, and retention data into an interactive computer program that realistically emulates a GLC instrument.

SIMULATION VS GOW-MAC DATA

GOW-MAC Column: SE-30 (DC-200)

Simulator Column: SE-30

Temperature: 100°

Solutes: 1. Acetone
2. 2-Butanone
3. 2-pentanone
4. 3-pentanone

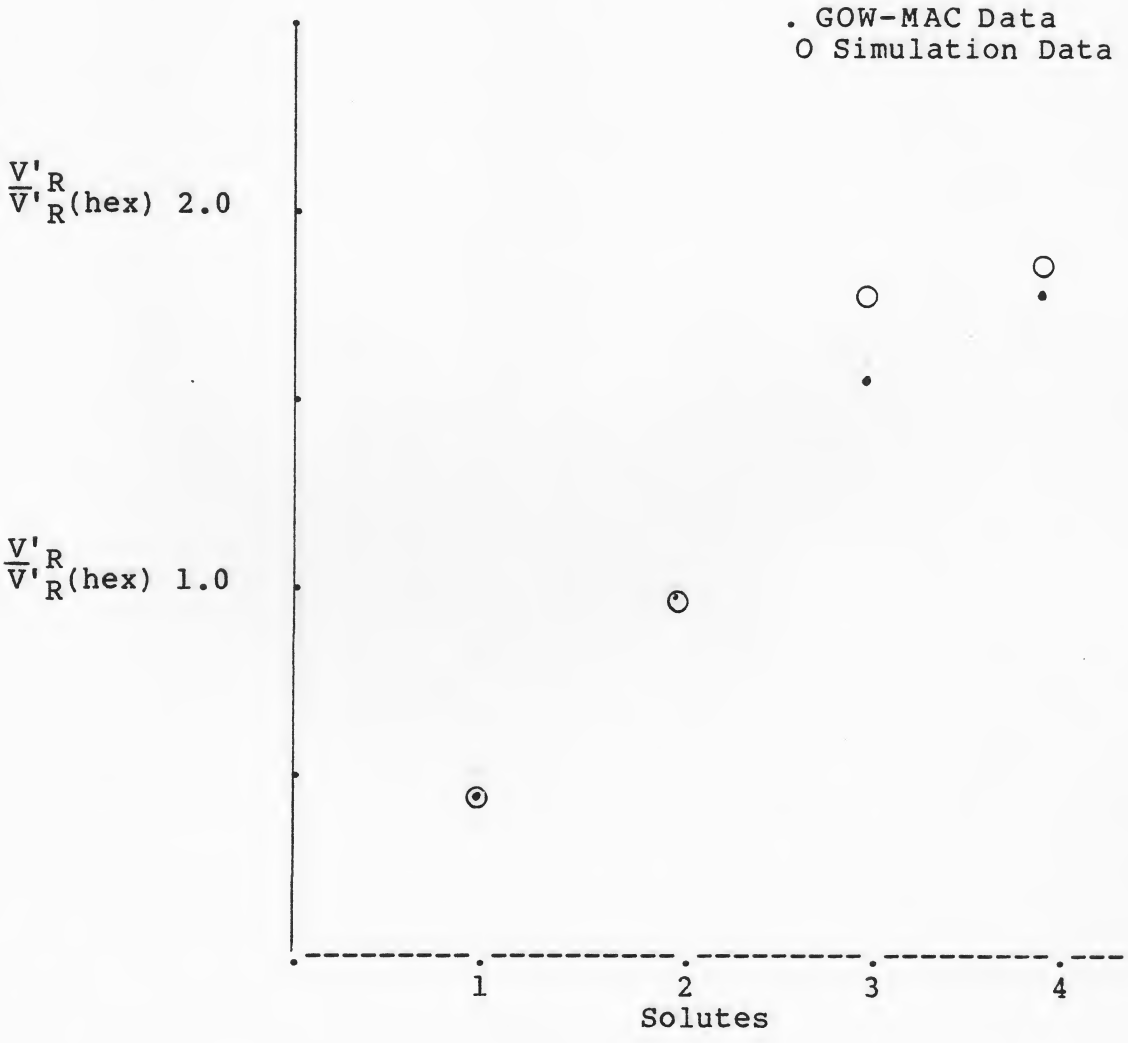

Figure 18. Comparison of simulator and GOW-MAC adjusted retention volumes for ketones.

SIMULATION VS GOW-MAC DATA

GOW-MAC Column: SE-30 (DC-200)

Simulator Column: SE-30

Temperature: 135°

Solutes: 1. Acetone
2. 2-Butanone
3. 2-pentanone
4. 3-pentanone

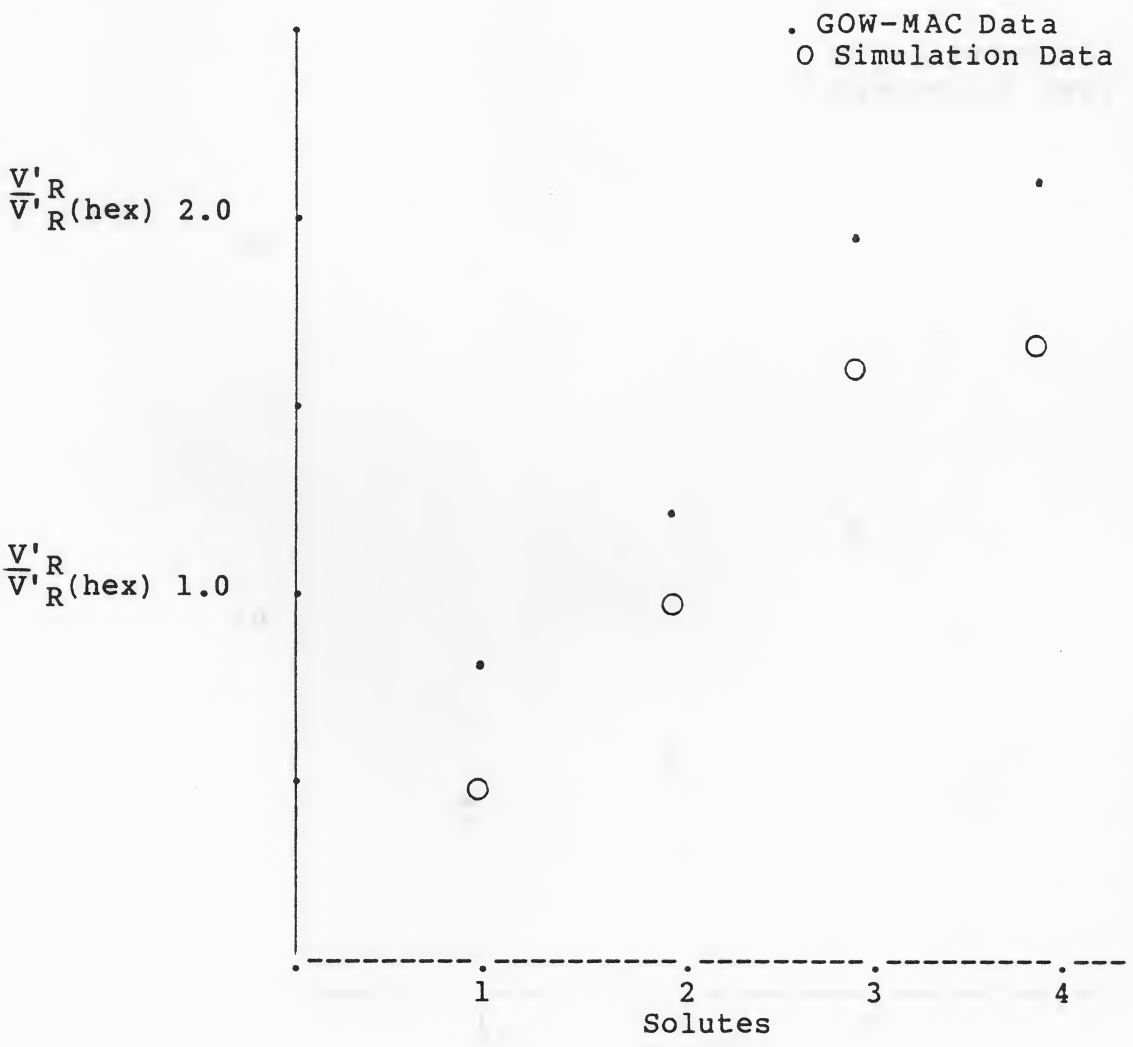

Figure 19. Comparison of simulator and GOW-MAC adjusted retention volumes for ketones.

SIMULATION VS SIGMA 300 DATA

Sigma 300 Column: Carbowax 20M/K-40M

Simulator Column: Carbowax 20M

Temperature: 135°

Solutes: 1. Methanol
2. Ethanol
3. Propanol

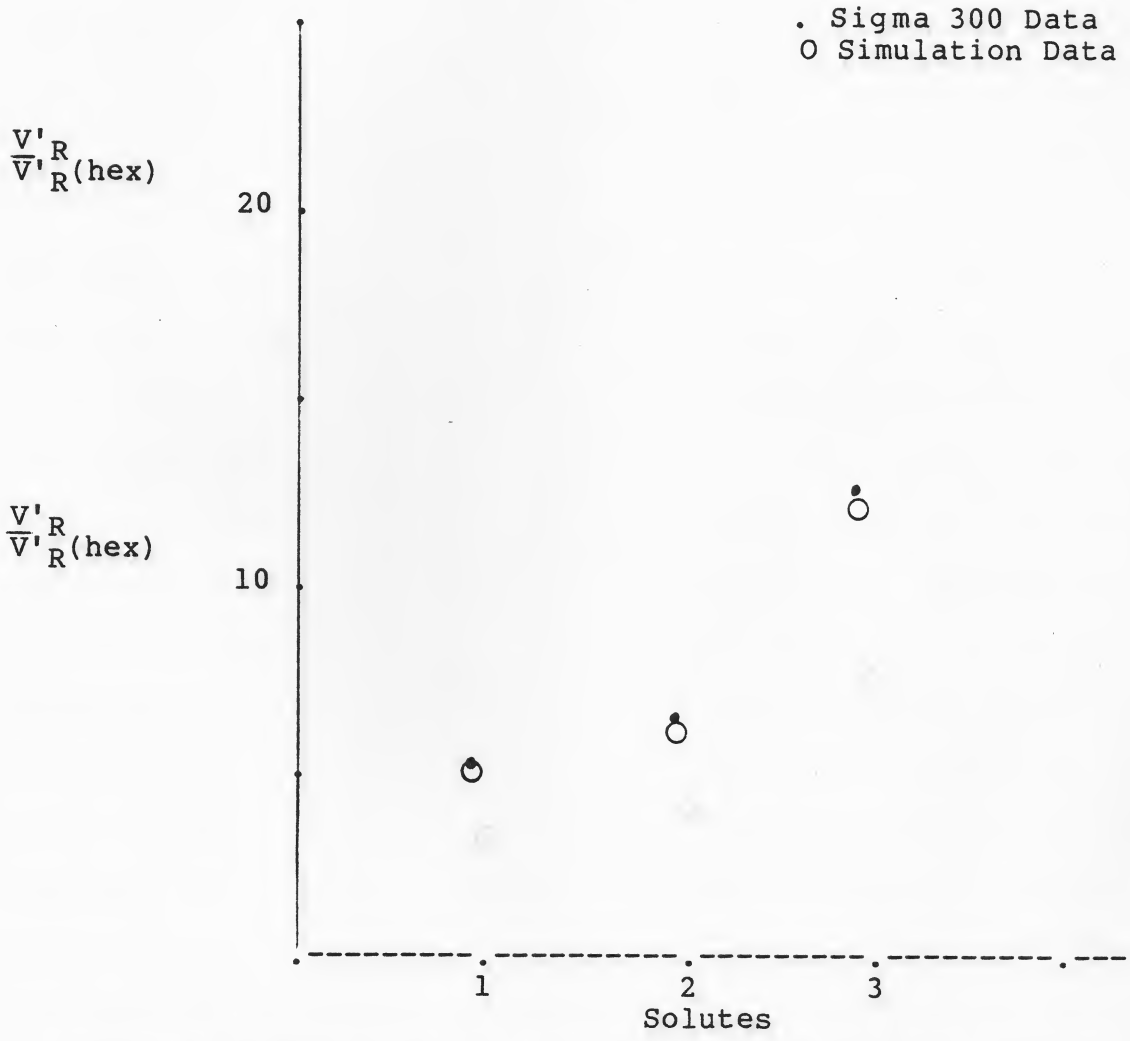

Figure 20. Comparison of simulator and Sigma 300 adjusted retention volume s for alcohols.

SIMULATION VS SIGMA 300 DATA

Sigma 300 Column: Carbowax 20M/K-40M

Simulator Column: Carbowax 20M

Temperature: 100°

Solutes: 1. Methanol
2. Ethanol
3. Propanol

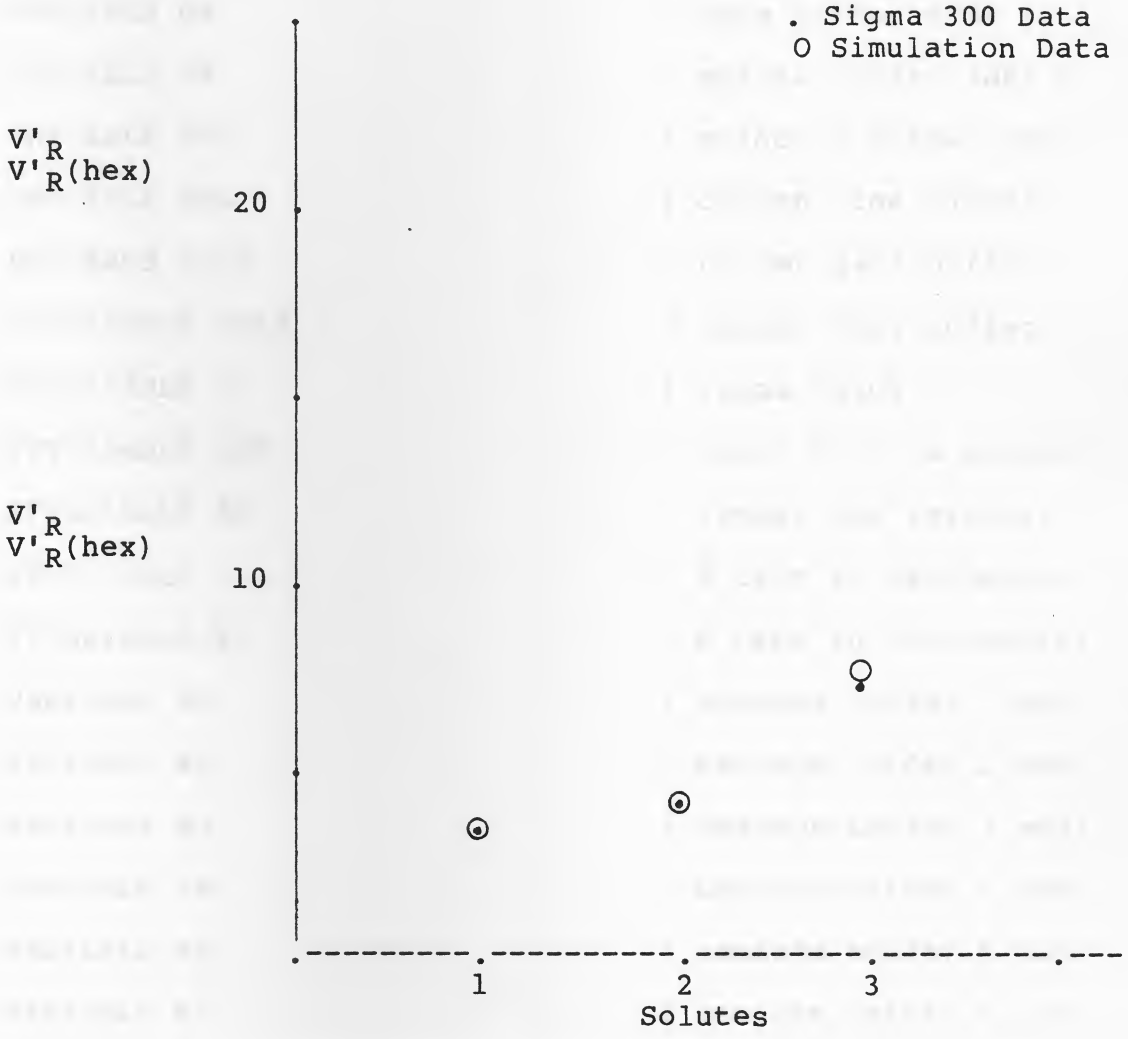

Figure 21. Comparison of simulator and Sigma 300 adjusted retention volume s for alcohols.

VI. APPENDIX: PROGRAM LISTING

```
SCR#1
VARIABLE STKRM                    ( stockroom buffer addr )
51300 CONSTANT GAC                ( gaussian curve buffer )
51900 CONSTANT GAC'               ( screen time buffer )
52100 CONSTANT GAC''              ( sigma markers buffer )
VARIABLE 2S                       ( counter index value )
VARIABLE DS                       ( data buffer addr )
VARIABLE PB                       ( points buffer addr )
VARIABLE PB2                      ( points 2 buffer addr )
VARIABLE COL                      ( column line number )
VARIABLE COL1                     ( column data buffer )
FPVARIABLE COL2                   ( column data buffer )
FPVARIABLE S                      ( sigma value )
FPVARIABLE SUM                    ( total ml in a sample )
FPVARIABLE MU                     ( linear gas velocity )
FPVARIABLE A                      ( A term in Van Deemter )
FPVARIABLE B                      ( B term in Van Deemter )
VARIABLE B1                       ( message buffer 1 addr )
VARIABLE B2                       ( message buffer 2 addr )
VARIABLE B3                       ( message buffer 3 addr )
VARIABLE B4                       ( message buffer 4 addr )
VARIABLE B6                       ( message buffer 6 addr )
VARIABLE B7                       ( message buffer 7 addr )
VARIABLE F                        ( flow rate )
```

```
SCR# 2
FPVARIABLE FL                    ( floating point flow )
FPVIABLE FL2                     ( column set number )
FPVARIABLE FL3                   ( air peak flow rate )
VARIABLE H                       ( height of peak )
FPVARIABLE U                     ( X position of peak )
VARIABLE ADD                     ( counter )
VARIABLE SPX                     ( sprite X position)
VARIABLE SPY                     ( sprite Y position )
FPVARIABLE ATN                   ( attenuation )
VARIABLE ATN'                    ( attenuation display )
FPVARIABLE N                     ( theoretical plates )
VARIABLE N2                      ( column set number )
VARIABLE O                       ( oven temp )
FPVARIABLE T                     ( oven temp K )
FPVARIABLE W                     ( liquid phase loading )
VARIABLE R                       ( scanning range )
VARIABLE RNG                     ( scanning range screen )
```

```
SCR# 3

      ( DATA CLEARING ROUTINES )
: PCLR PAD 64 0 FILL ;                          ( pad clear )
: DSCLR 929 0 DO 0 DS @ I + ! 0 DS @ I          ( data clear )
     + 6 + ! 0 DS @ I + 2+ ! 58 +LOOP ;
: PBCLR PB @ 1024 0 FILL                        ( point buffer
     PB2 @ 1024 0 FILL ;                          clear )
: SCLR GAC'' 200 0 FILL 0 2S ! ;                ( sigma clear )
: FCLR 1884 19 32 FILL                          ( function clear )
     1105 C@ 128 > IF -128 1105 +! -128
                   1106 +! THEN
     1305 C@ 128 > IF -128 1305 +! -128
                   1306 +! THEN
     1465 C@ 128 > IF -128 1465 +! -128
                   1466 +! THEN
     1665 C@ 128 > IF -128 1665 +! -128
                   1666 +! THEN ;
: CLR 147 EMIT ;                                ( clear screen )
```

```
SCR# 4
          ( STRING INPUT WORDS )
: INPUT PCLR PAD 1+ 64 EXPECT ;
: LEN 255 1 DO DUP I + C@ 0= IF I LEAVE  ( string length )
     THEN LOOP 1- SWAP C! ;
: NBRCK DUP 48 < IF DROP 1 ELSE          ( number check )
     DUP 57 > IF DROP 1 ELSE
     DROP 0 THEN THEN ;
: PADCK 3 1 DO PAD I + C@ NBRCK          ( pad check )
     1 = IF 0 LEAVE ELSE I
     2 = IF ." OK " THEN THEN LOOP ;
HEX
: DSTR$ SWAP OVER DABS <# #S ROT 0<       ( double number
     IF 2D ELSE 20 THEN HOLD #> >R          string )
     PAD 1+ R@ CMOVE R> PAD C! PAD ;
: STR$ S->D DSTR$ ;                      ( integer string )
DECIMAL
```

```
SCR# 5

          ( PROGRAM MESSAGES )

: REDO CR B3 @ 200 + 40 TYPE CR ;          ( repeat input )

: PAUSE CR B4 @ 640 + 40 TYPE BEGIN        ( " press stop key
     KEYIN 3 = IF 1 ELSE 0 THEN UNTIL ;      to continue " )


SCR# 6

          ( COLUMN GENERATOR )

: CODE 162 C@ INTFP FP .00196 FP*          ( generates column
     FP 2.5 FP+ FP .15 FPOVER FP/            set )
     FPSWAP FPDUP FP* FPOVER FP*
     FP 1000 FP* FPINT N2 !
     FP 1000 FP* FPINT FL2 ! ;
```

```
SCR# 7
        ( COLUMN INPUT / OUTPUT )
: CLMN1 CODE CLR CR CR CR B4 @ 760        ( prints column
    + 200 TYPE CR B4 @ 960 + 40 TYPE       set number )
    CR PCLR N2 @ STR$ 2 + 1412 3 CMOVE
    PCLR FL2 @ STR$ 2+ 1415 2 CMOVE
    PAUSE ;
: CLMNCK PAD 1+ C@ 88 = IF COL !         ( checks X next to
    ." OK " ELSE DROP THEN ;               column )
: NBRCK2 6 1 DO PAD I + C@ NBRCK         ( checks column
    1 = IF 0 REDO LEAVE ELSE I             number )
    5 = IF 1 THEN THEN LOOP ;
: PCK PAD C@ 5 = IF NBRCK2 ELSE 0 REDO   ( checks input
    THEN ;                                 length )
: CLMN2 CLR CR CR B3 @ 200 TYPE CR       ( column number
    BEGIN PCLR INPUT PAD LEN PCK UNTIL     input )
    PAD 4 + PAD 10 + 2 CMOVE PAD 4 +
    5 0 DO FILL PAD NUMBER DROP N2 !
    PAD 9 + NUMBER DROP FL2 ! ;
: COLUMNS CLR CR CR B4 @ 680 + 80 TYPE   ( combine column
    CR BEGIN KEYIN DUP 89 = IF DROP        input/output )
    CLMN2 1 ELSE
    78 = IF CLMN1 1 ELSE REDO 0 THEN
    THEN UNTIL ;
```

```
SCR# 8

          ( STOCKROOM DISPLAYS )

: STOCK CLR STKRM @ 999 TYPE 19 EMIT     ( print stockroom )
    CR CR ;

: MLSMPL 929 0 DO DS @ I + @ DUP 0> IF   ( prints amounts of
    DUP 9 > IF STR$ 2+ DS @ I + 4 + @      each chemical )
    13 + 1024 + 2 CMOVE ELSE STR$ 2+
    DS @ I + 4 + @ 14 + 1024 + 1 CMOVE
    THEN ELSE DROP THEN 58 +LOOP ;

: CLMNCLR 1891 1770 DO I 5 32 FILL 40    ( clears set
    +LOOP 1891 1770 DO I 1 15 FILL 40      number )
    +LOOP ;

: CRSCLR 214 C@ 16 > IF CLMNCLR ELSE     ( clears value next
    209 @ 211 C@ + 5 32 FILL THEN ;        to cursor )

: CLMNDSPL COL @ 40 * 1050 + 24 SWAP     ( column display )
    C! ;

: CLMNSET 1728 5 32 FILL PCLR N2 @ STR$  ( prints set
    2+ 1728 3 CMOVE PCLR FL2 @ STR$        number )
    2+ 1731 2 CMOVE ;
```

```
SCR# 9

          ( STOCKROOM CONTROLS )
: LLIM 214 C@ DUP 6 < IF DROP 5 214 C!    ( cursor line
    ELSE 21 > IF 21 214 C! THEN THEN ;      limit )
: ADJ DUP 11 > IF 3 - THEN 211 C@ 10 /    ( DS storage addr )
    2 AND 2 = IF 58 ELSE 0 THEN SWAP
    116 * 580 - DS @ + + ! ;
: VOLCK PAD C@ 2 > IF 0 ELSE PADCK        ( volume check )
    THEN ;
: MILCK VOLCK 1 = IF PAD NUMBER DROP      ( milliter check )
    SWAP ADJ ELSE DROP THEN ;
: STORE 214 C@ DUP 16 > IF CLMNCK ELSE    ( stores ml &
    MILCK THEN ;                            column )
: #ENTER CRSCLR PCLR PAD 1+ 3 EXPECT PAD  ( accepts ml
    LEN STORE ;                             amount )
: CRS 214 C@ 16 > IF 211 C@ DUP 26 = IF   ( controls cursor
    DROP 1 ELSE                             clearing )
    66 = IF 1 ELSE 0 THEN THEN ELSE
    211 C@ DUP 13 = IF DROP 1 ELSE DUP
    33 = IF DROP 1 ELSE DUP
    53 = IF DROP 1 ELSE
    73 = IF 1 ELSE 0 THEN THEN THEN
    THEN THEN IF #ENTER THEN ;
```

```
SCR# 10
      ( STOCKROOM CONTROLS CONT. )
: CRSR STOCK MLSMPL CLMNDSPL CLMNSET    ( combines all
    BEGIN KEYIN DUP 133 = IF DROP CRS     stockroom words )
    0 ELSE DUP
    17 = IF EMIT 0 ELSE DUP
    29 = IF EMIT 0 ELSE DUP
    145 = IF EMIT 0 ELSE DUP
    157 = IF EMIT 0 ELSE
    3 = IF 1 ELSE 0 THEN THEN THEN
    THEN THEN THEN LLIM UNTIL CLR ;


          ( SAMPLE DISPLAY )
: SMPLHDRCLR CRB2 @ 920 + 80 TYPE CR  ( sample makeup
    CR ."   COMPONENT " 3 SPACES        header )
    ." ML " CR ;
: % FP 0 SUM FP! 0 929 0 DO DS @ I + @  ( total volume in
    + DS @ I + 6 + @ + 58 +LOOP INTFP     sample )
    SUM FP! ;
: PERCENT 929 0 DO DS @ I + DUP 6 + @   ( percent of each
    SWAP @ + INTFP SUM FP@ FP/ FP 100     component )
    FP* FP .5 FP+ FPINT DS @ I + 2+ !
    58 +LOOP ;
```

```
SCR# 11

        ( SAMPLE DISPLAY CONT. )
: DISPLAY PERCENT SMPLHDR CR 929 0 DO  ( chemicals in
    DS @ I + @ 0> IF DS @ I 4 + + @      sample )
    STRRM @ + 11 TYPE 4 SPACES DS @
    I + ? CR THEN 58 +LOOP ;
: DSPLY % SUM FP@ FP 0 FP= IF CR       ( combines chemical
    ." NOTHING IN SAMPLE " CR ELSE       and volume )
    DISPLAY THEN ;

          ( SAMPLE CONTROL )
: SAMPLE CRSR DSPLY PAUSE ;            ( stockroom words )
```

```
SCR# 12
      ( HIGH RESOLUTION ON / OFF )
HEX
: 3BNK E000 C400 VIDMEM ;                ( VICII bank 3 )
: 0BNK CLRBM 1000 400 VIDMEM ;           ( VICII bank 0 )
: HIRES 3BNK SETBM CLRBMM 10 FILSCRN     ( high resolution
    0 CURX ! 0 CURY ! ;                    on )
DECIMAL

SCR# 13
      ( GC CHROMATOGRAM FORMAT )
: LINES 0 88 319 88 LINE                 ( draws screen
    0 176 319 176 LINE ;                   lines )
: TSCLR GAC' 31 + GAC' DO I @ I 2+ @ I   ( clears screen
    4 + @ RCHR 6 +LOOP ;                   time )
: TSDSPL RNG @ 0= IF GAC' 55 + GAC' 36   ( displays screen
    + ELSE GAC' 91 + GAC' 60 + THEN        time )
    DO I @ I 2+ @ I 4 + @ RCHR
    6 +LOOP ;
: AXIS LINES TSCLR TSDSPL ;              ( combines hires
                                           screen words )
```

```
SCR# 14
    ( GC CHROMATOGRAM FORMAT CONT. )
2 BASE !
SPDEFINE SPR1                          ( sprite shape )
00000000 00000000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00000000 00000000
01111111 10000001 11111110
01111111 10000001 11111110
00000000 00000000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00011000 00000000
00000000 00000000 00000000
DECIMAL
```

```
SCR# 15
   ( GC CHROMATOGRRAM FORMAT CONT. )
HEX
: SPRT SPR1 C200 SPMOV 1 C200 SPPTR 1     ( sprite on )
     SPMCOFF 1 SP<BKGD 1 1 SPCOLOR
     1 17 31 SPXY 1 SPON ;

      ( GC CHROMATOGRAM DISPLAY )
: HCLR FF40 18 0 FILL ;                   ( clears height )
: TR#CLR FDOO 28 0 FILL ;                 ( clears time )
DECIMAL
: TR' 20 8 184 RCHR 18 16 184 RCHR        ( prints " TR " )
     13 111 184 RCHR 9 119 184 RCHR
     14 127 184 RCHR ;
: HEIGHT 8 8 192 RCHR 5 16 192 RCHR       ( " HEIGHT " )
     9 24 192 RCHR 7 32 192 RCHR
     8 40 192 RCHR 20 48 192 RCHR ;
: HDRS TR' HEIGHT ;
: HGHTADJ SPY@ DUP 126 < IF INTFP FP -1   ( adjusts displayed
     FP* FP 126 FP+ FPINT ELSE DUP          height value )
     214 > IF DROP 0 ELSE DUP
     129 > IF INTFP FP -1 FP* FP 214
     FP+ FPINT ELSE DROP 0
     THEN THEN THEN ;
```

```
SCR# 16
   ( GC CHROMATOGRAM DISPLAY CONT. )
: HGHT HCLR HGHTADJ STR$ C@ DUP          ( displays height
     2 = IF DROP 48 48 PAD 2+ C@           position of
     ELSE DUP                              sprite )
     3 = IF DROP 48 PAD 2+ C@ PAD 3 + C@
     ELSE DUP
     4 = IF DROP PAD 2+ C@ PAD 3 + C@
     PAD 4 + C@ ELSE DROP THEN THEN THEN
     80 192 RCHR 72 192 RCHR
     64 192 RCHR ;
```

```
SCR# 17
   ( GC CHROMATOGRAM DISPLAY CONT. )
: TR# TR#CLR RNG @ 0= IF SPY @ 129 < IF   ( displays time
    SPX @ ELSE SPX @ 319 + THEN ELSE        position of
    SPY @ 129 < IF SPX @ 638 + ELSE         cross )
    SPX @ 957 + THEN THEN INTFP
    FP 1.5674 FP* FP -20.376 FP+ FPINT
    STR$ C@ DUP
    2 = IF DROP 48 48 48 PAD 2+ C@
    ELSE DUP
    3 = IF DROP 48 48 PAD 2+ C@ PAD 3 +
    C@ ELSE
    4 = IF 48 PAD 2+ C@ PAD 3 + C@ PAD
    4 + C@ ELSE PAD 2+ C@ PAD 3 + C@
    PAD 4 + C@ PAD 5 + C@
    THEN THEN THEN
    96 184 RCHR 88 184 RCHR
    46 80 184 RCHR 72 184 RCHR
    64 184 RCHR ;
```

```
SCR# 18
        ( SPRITE CROSS CONTROLS )
: LIMY 41 < IF DROP 41 41 ELSE DUP        ( limits cross Y
     217 > IF DROP 217 217 ELSE DUP         movement )
     THEN THEN ;
: LIMX 13 < IF DROP 13 13 ELSE DUP        ( limits cross X
     332 > IF DROP 332 332 ELSE DUP         movement )
     THEN THEN ;
: CRMOV 135 = IF 1 SPX @ SPY @ 8 + DUP    ( controls cross
     LIMY SPY ! SPXY DROP 0 ELSE DUP        fast scan )
     136 = IF 1 SPX @ 8 + DUP LIMX
     SPX ! SPY @ SPXY DROP 0 ELSE DUP
     139 = IF 1 SPX @ SPY @ 8 - DUP
     LIMY SPY ! SPXY DROP 0 ELSE DUP
     140 = IF 1 SPX @ 8 - DUP LIMX
     SPX ! SPY @ SPXY DROP 0 ELSE
     THEN THEN THEN THEN ;
```

```
SCR# 19
    ( SPRITE CROSS CONTROLS CONT. )
: CROSS 23 SPX ! 49 SPY ! TR# HGHT SPRT  ( combines sprite
    BEGIN KEYIN DUP 3 = IF DROP 1 ELSE    control words )
    DUP
    17 = IF 1 SPX @ SPY @ 1+ DUP LIMY
    SPY ! SPXY DROP 0 ELSE DUP
    29 = IF 1 SPX @ 1+ DUP LIMX SPX !
    SPY @ SPXY DROP 0 ELSE DUP
    145 = IF 1 SPX @ SPY @ 1- DUP LIMY
    SPY ! SPXY DROP 0 ELSE DUP
    157 = IF 1 SPX @ 1- DUP LIMX SPX !
    SPY @ SPXY DROP 0 ELSE DUP
    133 = IF TR# HGHT DROP 0 ELSE DUP
    CRMOV DROP 0 THEN THEN THEN THEN
    THEN THEN UNTIL ;
```

```
SCR# 20
    ( INSTRUMENT CONTROLS DISPLAYS )
: INSTDSPLY CLR B6 @ 993 TYPE                ( instrument
    33 0 DO 157 EMIT LOOP                      screen )
    3 0 DO 145 EMIT LOOP ;
: P$CLR PAD 40 32 FILL ;                     ( clears pad )
: OCLR 1396 10 32 FILL ;                     ( clears temp )
: FLCLR 1596 10 32 FILL ;                    ( clears flow )
: F1 FCLR 1109 1884 19 CMOVE 128 1105 +!     ( displays
    128 1106 +! ;                              attenuation )
: F2 FCLR 1309 1884 16 CMOVE 128 1305 +!     ( displays temp )
    128 1306 +! ;
: F3 FCLR 1509 1884 9 CMOVE 128 1465 +!      ( displays flow )
    128 1466 +! ;
: F4 FCLR 32 R @ C! 1679 1884 15 CMOVE       ( displays range )
    128 1665 +! 128 1666 +! ;
: FLDSPLY FLCLR P$CLR F @ STR$ 1+ 1596       ( flow value to a
    5 CMOVE ;                                  string )
: ODSPLY OCLR P$CLR 0 @ STR$ 1+ 1396         ( temp value to a
    5 CMOVE ;                                  string )

    ( INSTRUMENT CONTROLS )
: ORANGE O @ 200 > IF 200 O ! ELSE           ( checks temp
    O @ 80 < IF 80 O ! THEN THEN ;             range )
```

```
SCR# 21
        ( INSTRUMENT CONTROLS )
: ATTEN 32 ATN @ C! ATN @ DUP                ( attenuation
     1258 = IF DROP 1230 ELSE 4 + THEN         entry )
     DUP ATN ! 40 - PCLR PAD 1+ 3 CMOVE
     PAD LEN PAD FPVAL FP 16 FPSWAP FP/
     ATN' FP! 30 ATN @ C! ;
: OTEMP BEGIN KEYIN DUP                      ( temp entry )
     145 = IF DROP 1 0 +! 0 ORANGE
     ELSE DUP
     17 = IF DROP -1 0 +! 0 ORANGE ELSE
     134 = IF 1 ELSE 0 THEN THEN THEN
     ODSPLY UNTIL O @ 273 + INTFP T FP!
     FCLR ;
: FLRANGE F @ 1 < IF 1 F ! ELSE              ( checks flow
     F @ 125 > IF 125 F ! THEN THEN ;          range )
: FLOW BEGIN KEYIN DUP                       ( flow entry )
     145 = IF DROP 1 F +! 0 FLRANGE
     ELSE DUP
     17 = IF DROP -1 F +! 0 FLRANGE ELSE
     135 = IF 1 ELSE 0 THEN THEN THEN
     FLDSPLY UNTIL F @ INTFP FL FP!
     FCLR ;
: RANGE R @ 1764 = IF 31 1804 DUP R ! C! ( screen range
     640 RNG ! ELSE                            entry )
     31 1764 DUP R ! C! 0 RNG ! THEN ;
```

```
SCR# 22

     ( INSTRUMENT CONTROLS CONT. )
: INST INSTDSPLY 30 ATN @ C! ODSPLY      ( combines inst
     31 R @ C! FLDSPLY BEGIN KEYIN DUP     words )
     133 = IF DROP FCLR F1 ATTEN 0
     ELSE DUP
     134 = IF DROP FCLR F2 OTEMP 0
     ELSE DUP
     135 = IF DROP FCLR F3 FLOW 0
     ELSE DUP
     136 = IF DROP FCLR F4 RANGE 0 ELSE
     3 = IF 1 ELSE 0 THEN THEN THEN THEN
     THEN UNTIL ;

          ( STARTUP DATA )
: STARTUP 1230 ATN ! 1764 R ! 80 O !     ( fills startup
     50 F ! FP 353 T FP ! FP 50 FL FP!     variables )
     FP 16 ATN' FP! 0 RNG ! 18 COL !
     DSCLR 144 EMIT 1 53281 C! ;
```

```
SCR# 23

        ( CURVE CALCULATIONS )

: HGHTCK SWAP OVER @ + DUP 83 > IF DROP   ( checks peak
    83 THEN SWAP ! ;                        height )
: FPCK FPDUP FP 0 FP< IF FPDROP FP 0      ( checks peak
    ELSE FPDUP FP 300 FP> IF FPDROP         width )
    FP 310 THEN THEN ;
: POINTS 2* 20 - GAC + @ H @ 1000 */ ;    ( selects points )
: DP1 320 0 DO                            ( calculates top
    I INTFP U FP@ FP- FPABS                 half of screen
    S FP@ FP/ FP 100 FP* FPCK FPINT         points )
    DUP 10 < IF DROP H @ PB @ I 2* +
    HGHTCK ELSE DUP
    300 > IF DROP ELSE POINTS
    PB @ I 2* + HGHTCK THEN THEN LOOP ;
: DP2 640 320 DO                          ( calculates bottom
    I INTFP U FP@ FP- FPABS                 half of screen
    S FP@ FP/ FP 100 FP* FPINT DUP          points )
    10 < IF DROP H @ PB2 @ I 320 -
    2* + HGHTCK ELSE DUP
    300 > IF DROP ELSE POINTS PB2 @
    I 320 - 2* + HGHTCK THEN THEN
    LOOP ;
: AIR FL FP@ T FP@ FP* FP .00211 FP*      ( air peak time )
    FL3 FP! ;
```

```
SCR# 24
      ( CURVE CALCULATIONS CONT. )
: AIRPLOT FP 20.96 FL3 FP@ FP/ FP 64 FP* ( air retention
    U FP! 7 H ! ;                          time )
: COLADJ COL @ DUP 18 = IF DROP FP 10     ( column
    COL2 FP! 10 COLI ! FP 2.8032 W FP!      parameters )
    ELSE DUP
    19 = IF DROP FP 8 COL2 FP! 22
    COLI ! FP 2.56 W FP! ELSE DUP
    20 = IF DROP FP 4 COL2 FP!
    34 COLI ! FP 2.8928 W FP! ELSE DUP
    21 = IF DROP FP 6 COL2 FP!
    46 COLI ! FP 3.2384 W FP! ELSE DROP
    THEN THEN THEN THEN ;
: RETTIME FP 10 B FP@ T FP@ FP/ A FP@     ( retention times )
    FP+ FP1 W FP@ T FP@ FP* FP*
    FP 273 FP/ FP .65 FP/ FP 20.96 FP+
    FL3 FP@ FP/ FP 64 FP* U FP! ;
: CURSEN H @ INTFP ATN' FP@ FP* FP .5 ( adjust peak
    FP+ FPINT H ! ;                        height based on
                                           attenuator
                                           setting )
```

```
SCR# 25
     ( CURVE CALCULATIONS CONT. )
: VAND FP 122 FPDUP FP 20.96          ( theoretical
    FL3 FP@ FP/ FP 60 FP* FP/ MU FP!    plates based on
    FP .08 N2 @ INTFP FP 1000 FP/ MU    van Deemter )
    FP@ FP/ FP+ FL2 @ INTFP FP 1000
    FP/ MU FP@ FP* FP+ FP/ FP 2.5 FP*
    N FP! ;
: SIGMA U FP@ N FP@ FP 16 FP/ FPSQR FP/  ( calculates peak
    FP 4 FP/ S FP! ;                      width )
: AREA N FP@ FPSQR U FP@ FP 64 FP/ FP/   ( calculates peak
    H @ INTFP FP 25.07 FP/ FP* FP .5      height )
    FP+ FPINT H ! ;

     ( CURVE PLOTTING ROUTINES )
: PLT1 0 CURX ! 85 CURY !             ( plots top of
    320 0 DO I PB @ I 2* + @            hires screen )
    85 SWAP - LINETO LOOP ;
: PLT2 0 CURX ! 173 CURY !            ( plots bottom of
    320 0 DO I I 2* PB2 @ + @           hires screen )
    173 SWAP - LINETO LOOP ;
```

```
SCR# 26
   ( CURVE PLOTTING ROUTINES CONT. )
: DATAPOINTS U FP@ RNG @ INTFP FP-        ( controls
     FPDUP U FP! FPINT DUP                  plotting )
     640 > IF DROP ELSE DUP
     350 > IF DROP DP2 ELSE
     280 < IF DP1 ELSE DP1 DP2
     THEN THEN THEN ;
: SADD 2S @ 4 + 2S ! ;                    ( counter )
: MARK U FP@ S FP@ FP 2 FP* FPOVER        ( calculates 4s for
     FPOVER  FP- FPINT GAC'' 2S @ + !       peaks )
     FP+ FPINT GAC'' 2S @ + 2+ ! SADD ;
: MARK2 GAC'' 65 + GAC'' DO I @ DUP 0>    ( plots 4s under
     IF DUP 319 < IF 101 SWAP 84 RCHR       each peak )
     ELSE 319 - 101 SWAP 172 RCHR THEN
     ELSE DROP THEN 2 +LOOP ;
: PEAKS PCCLR SCLR COLADJ AIR VAND RNG @ ( combines 4s
     0= IF AIRPLOT SIGMA AREA CURSEN        words )
     DATAPOINTS THEN
     929 0 DO I 2+ DS @ + @ 0> IF I DS @
     + COLI @ + DUP FP@ A FP! 6 + FP@ B
     FP! I DS @ 2+ + @ H ! RETTIME SIGMA
     AREA CURSEN DATAPOINTS MARK THEN
     58 +LOOP ;
```

```
SCR# 27
   ( CURVE PLOTTING ROUTINES CONT. )
: DELAY 4 ZOOM ! HIRES 7 94 40 RCHR        ( displays enlarged
    1 134 40 RCHR 19 174 40 RCHR             letters )
    2 ZOOM ! 8 16 1 18 7 15 20 1 13 15
    18 8 3 320 0 DO I 104 RCHR 24 +LOOP
    0 ZOOM ! ;
: GC DELAY PEAKS HIRES AXIS HDRS PLT1      ( combines
    PLT2 MARK2 CROSS 0BNK 1 SPOFF ;          chromatogram
                                             words )
        ( UNKNOWN EXPERIMENT )
     ( RANDOM SAMPLE GENERATORS )
: ?#DS PAD C@ DUP 2 = IF DROP PAD 2+       ( adjust memory )
    PAD 4 + 1 CMOVE PAD 2+ 2 48 FILL
    ELSE 3 = IF PAD 2+ PAD 3 + 2 <CMOVE
    48 PAD 2+ C! THEN THEN ;
: RNDM1 0 162 C@ 0 DO 58 + DUP 870 > IF    ( 1st random word )
    DROP 0 THEN LOOP ;
: RNDM2 BEGIN RNDM1 RNDM1 DUP ROT DUP      ( selects & prints
    ROT = IF DROP DROP 0 ELSE 1 THEN         code for random
    UNTIL DUP PCLR STR$ 2+ ?#DS 1167         chemicals )
    3 CMOVE OVER PCLR STR$ 2+ ?#DS
    1170 3 CMOVE ;
: RNDM3 162 C@ 7 AND DUP 0= IF DROP 1      ( generates random
    THEN 56324 C@ 7 AND DUP 0= IF DROP       volumes )
    1 THEN ;
```

```
SRC# 28
   ( RANDOM SAMPLE GENERATORS CONT. )
: RNDM4 BEGIN RNDM3 DUP 3 PICK = IF      ( displays random
     DROP DROP 0 ELSE 1 THEN UNTIL         volumes )
     PCLR DUP STR$ 2+ 1173 1 CMOVE
     PCLR OVER STR$ 2+ 1174 1 CMOVE ;
: RNDM DSCLR RNDM2 RNDM4 ROT DS @ + 6    ( combines random
     + ! SWAP DS @ + 6 + ! ;               sample words )

   ( UNKNOWN EXPERIMENT MESSAGES )
: MSG1 CR B3 @ 240 + 40 TYPE CR ;
: MSG2 CR B3 @ 280 + 80 TYPE CR ;
: MSG3 CR B3 @ 360 + 80 TYPE CR ;
: MSG4 CR B3 @ 440 + 80 TYPE CR ;
: MSG5 CR B3 @ 520 + 80 TYPE CR ;
: MSG6 CR B3 @ 600 + 80 TYPE CR ;
: MSG7 CR B3 @ 680 + 40 TYPE CR ;
: MSG8 CR B3 @ 720 + 80 TYPE CR ;
: MSG9 CR B3 @ 800 + 80 TYPE CR ;
: MSG10 CR B3 @ 880 + 80 TYPE CR ;
```

```
SCR# 29

          ( UNKNOWN EXPERIMENT )

: ?NBRCK BEGIN INPUT PAD LEN 20 1 DO PAD ( checks unknown
    I + C@ DUP 0 = IF DROP 1 LEAVE ELSE   code number
    NBRCK IF 0 REDO THEN THEN LOOP        input )
    UNTIL ;

: ?% INTFP FPDUP PAD NUMBER DROP INTFP    ( checks volume in
    FP- FPABS FPSWAP FP/ FP .1 FP> IF       answer )
    0 ELSE 1 THEN ;

: ?IN CLR CR CR MSG7 BEGIN PCLR 1 INPUT   ( accepts unknown
    PAD LEN PAD C@ 8 = IF 9 1 DO I PAD      input )
    + C@ NBRCK IF DROP 0 REDO LEAVE
    THEN LOOP ELSE DROP 0 REDO THEN
    UNTIL ;

: ?DECODE PAD 8 = PAD 51 + 1 CMOVE PAD    ( decodes unknown
    50 + NUMBER DROP PAD 4 + PAD 41 +       number to
    3 CMOVE PAD 40 + NUMBER DROP DS @ +   chemicals and
    6 + ! PAD 7 + PAD 31 + 1 CMOVE PAD    volume )
    30 + NUMBER DROP PAD 1+  PAD 21 + 3
    CMOVE PAD 20 + NUMBER DROP DS @ +
    6 + ! ;
```

```
SCR# 30
      ( UNKNOWN EXPERIMENT CONT. )
: UNKNOWN CLR DSCLR CR CR BEGIN MSG6      ( combines unknown
    KEYIN DUP 89 = IF DROP ?IN ?DECODE     random & answer
    % PERCENT 1 ELSE                       input )
    78 = IF 1 CLR CR CR MSG8 RNDM %
    PERCENT PAUSE ELSE 0
    REDO THEN THEN UNTIL ;

          ( UNKNOWN COMPARE )
: PCLR2 PAD 30 + 34 0 FILL ;              ( clears pad )
: PADJ PAD 64 + PAD 30 + DO I C@ 32 =     ( adjust string in
    IF 0 I C! THEN LOOP ;                  pad )
: PMOVE PAD 40 + PAD 30 + DO I C@ 0> IF   ( moves answer
    I PAD 46 + 12 CMOVE LEAVE THEN LOOP    string in pad )
    PAD 45 + LEN ;
: PEQUIL 0 12 0 DO PAD I + C@ PAD 45 + I  ( compares strings
    + C@ = IF DROP 1 ELSE DROP 0 LEAVE     for equality )
    THEN LOOP ;
```

```
SCR# 31

        ( UNKNOWN COMPARE CONT. )
: PCOMPARE CR MSG9 PCLR INPUT PAD LEN     ( compares answer
    986 0 DO DS @ I + 6 + @ 0> IF           to values in DS
    DS @ I + 4 + @ STKRM @ + PAD 30 +       buffer )
    12 CMOVE PADJ PMOVE PEQUIL IF MSG1
    MSG2 ?NBRCK DS @ I + 2+ @ ?% IF
    MSG3 LEAVE ELSE MSG4 LEAVE THEN
    ELSE PCLR2 THEN THEN I 928 = IF
    MSG5 THEN 58 +LOOP ;
: PCMPR DSCLR ?IN ?DECODE % PERCENT        ( combines answer
    BEGIN PCOMPARE CR BEGIN MSG10 KEYIN     input and check
    CLR CR CR DUP                           words )
    89 = IF DROP 0 1 ELSE
    78 = IF 1 1 ELSE REDO 0 THEN THEN
    UNTIL UNTIL ;

        ( HETP EXPERIMENT )
: HETP CLR B2 @ 880 TYPE PAUSE            ( directions for
    CLR B1 @ 360 TYPE DSCLR PAUSE ;         HETP exp )
```

```
SCR# 32

            ( DEMONSTRATION )
: DEMO CLR CR CR B1 @ 360 + 640 TYPE      ( controls
     PAUSE CLR B7 @ 880 TYPE PAUSE          demonstration )
     1246 ATN ! 1764 R ! 120 O ! 50 F !
     FP 393 T FP! FP 50 FL FP! FP 1 ATN'
     FP! 0 RNG ! 21 COL ! DSCLR 161 @
     15 AND 162 C@ 15 AND 56324 C@ 15
     AND 56325 C@ 15 AND DS @ ! 116 DS @
     + ! 232 DS @ + ! 348 DS @ +!
     SAMPLE INST GC ;

       ( NEW COLUMNS / RESTART )
: NEW DSCLR PBCLR STARTUP COLUMNS ;

          ( SAMPLE CLEAR )
: SAMPLCLR CLR CR CR CR B2 @ 880 +
     40 TYPE CR CR DSCLR PAUSE ;

         ( PREVIOUS SCREEN )
: PRV 3BNK AXIS HDRS SETBM CROSS 0BNK
     1 SPOFF ;

          ( MENUE DISPLAY )
: MENDSPLY CLR CR CR CR B4 @ 80 TYPE CR
     CR B4 @ 80 + 560 TYPE ;
```

```
SCR# 33

                ( BLOCK LOAD )
: BLOAD 89 BLOCK GAC 800 CMOVE          ( loads message
    EMPTY-BUFFERS 75 BLOCK DS ! 40        buffers )
    BLOCK STKRM ! 41 BLOCK PB ! 42
    BLOCK PB2 ! 43 BLOCK B1 ! 44 BLOCK
    B2 ! 45 BLOCK B3 ! 46 BLOCK B4 !
    47 BLOCK B7 ! 48 BLOCK B6 ! ;

                ( MAIN MENU )
: MENU BLOAD STARTUP COLUMNS BEGIN      ( combines all
    MENDSPLY KEYIN DUP                    simulators
    133 = IF DROP DEMO 0 ELSE DUP         operations )
    137 = IF DROP SAMPLE 0 ELSE DUP
    134 = IF DROP INST 0 ELSE DUP
    138 = IF DROP GC 0 ELSE DUP
    135 = IF DROP PRV 0 ELSE DUP
    139 = IF DROP HETP 0 ELSE DUP
    136 = IF DROP UNKNOWN 0 ELSE DUP
    140 = IF DROP PCMPR O ELSE DUP
    144 = IF DROP SMPLCLR 0 ELSE
    5 = IF NEW 0 ELSE 0 THEN THEN THEN
    THEN THEN THEN THEN THEN THEN THEN
    UNTIL ;
```

SCR# 43

*********HETP EXPERIMENT CONT.********* (B1 message buffer)

* THE HETP IS THEN CALCULATED FOR EACH CHROMATOGRAM.

HETP = L/N

(L = 4 FT FOR ALL COLUMNS)

* PLOT THE HETP VERSUS GAS VELOCITY.

****GC INTRODUCTION & DEMONSTRATION****

THIS COMPUTER PROGRAM SIMULATES THE OPERATION OF A BASIC GAS CHROMATOGRAPH. TO ACQUAINT YOU WITH ITS FEATURES, THE COMPUTER WILL PERFORM A DEMONSTRATION. THE DEMO CONSISTS OF A DIAGRAM OF A GC SYSTEM, AND THEN STEPS THROUGH THE STOCK ROOM, INSTRUMENT CONTROLS, AND ENDS WITH THE GENERATION OF A CHROMATOGRAM. ALL THE OPERATOR NEED DO TO STEP THROUGH THE DEMO IS PRESS THE (STOP) KEY AFTER STUDYING EACH SCREEN. ONCE THE DEMO RETURNS TO THE MENUE THE SIMULATOR IS UNDER THE OPERATORS FULL CONTROL.

scr# 44

**********HETP EXPERIMENT********** (B2 message buffer)

* THE PURPOSE OF THIS EXPERIMENT IS TO DETERMINE THE OPTIMUM FLOW RATE FOR THE COLUMN.

* TO DO THIS, SELECT A COLUMN AND ONE CHEMICAL FROM THE STOCKROOM.

* RUN A NUMBER OF CHROMATOGRAMS KEEPING TEMPERATURE CONSTANT, BUT VARYING THE FLOW RATE.

* USING THE CROSS, MEASURE THE RETENTION TIME 'X', AND THE PEAK WIDTH AT THE POINTS OF INFLECTION 'Y'.

* CALCULATE THE NUMBER OF THEORETICAL PLATES 'N' BY

$$N = 16 \left(\frac{X}{Y}\right)^2$$

THE SAMPLE IS EMPTIED

THESE ARE THE COMPONENTS ACKNOWLEDGED

TO BE IN THE SAMPLE.

SCR# 75

(DATA STORAGE BUFFER CONTENTS)

0	0	200	0	0
-1.0464	578.0238	-2.6180	1221.9909	-1.5137
869.5887	-3.3068	1752.4244	0	0
220	0	0	-0.2428	388.3321
-2.1998	1063.0350	-0.7026	445.6614	-2.4015
943.7945	0	0	240	0
0	-1.4279	845.9218	-2.9865	1441.4104
-1.5672	970.8228	-3.4863	1852.6938	0
0	260	0	0	-1.1411
1003.1726	-2.4681	1392.6586	-1.511	873.6449
-2.7535	1280.6493	0	0	280
0	0	-14155	994.9687	-2.9041
1527.3601	-1.7154	1171.2282	-3.5983	2000.2192
0	0	300	0	0
-1.4277	1367.2766	-2.9099	1772.9581	-1.5957
1289.2657	-2.7111	1456.7431	0	0
320	0	0	-1.5804	1202.9442
-3.1543	1742.1982	-1.7668	1322.6794	-3.7201
2149.0217	0	0	340	0
0	-1.8065	1759.6818	-3.4545	2192.3111
-1.7569	1580.5533	-3.3370	1897.5491	0
0	480	0	0	-1.3314
1157.5885	-2.5475	1505.4949	-1.3825	1133.2559
-2.7996	1648.9341	0	0	500

(DATA STORAGE BUFFER CONT.)

0	0	-1.4331	906.0156	-2.5105
1280.6493	-1.2718	894.5169	-2.7213	1477.1963
0	0	520	0	0
-1.4547	1344.5120	-2.7454	1692.9263	-1.5262
1317.9374	-3.0238	1829.5152	0	0
540	0	0	-1.3574	1025.0559
-2.7422	1490.0339	-1.5240	1118.5508	-2.9236
1642.3044	0	0	560	0
0	-1.5857	1555.0615	-2.9403	1900.4313
-1.6557	1524.2003	-3.4703	2142.8733	0
0	580	0	0	-1.3000
1115.6043	-2.7986	1611.9972	-1.5335	1227.9721
-3.1056	1790.4558	0	0	600
0	0	-1.4585	1477.6833	-3.0588
1920.5193	-1.5586	1449.7431	-3.3275	2044.6213
0	0	620	0	0
-1.2832	1123.7884	-2.8408	1636.8692	-1.5402
1236.1692	-3.1640	1812.1668		

SCR# 89

(GAUSSIAN CURVE DATA BUFFER CONTENTS)

995	994	993	992	990	989	987	986	984	982	980	978
974	972	969	967	964	962	959	956	953	950	947	944
941	937	934	930	927	923	919	916	912	908	904	899
895	891	887	882	878	874	869	864	860	855	850	845
840	835	830	825	820	815	810	804	799	794	788	783
777	772	766	760	755	749	743	738	732	726	720	714
709	703	697	691	685	679	673	667	661	655	649	642
637	631	625	619	613	607	600	594	588	582	576	570
564	558	552	546	540	534	528	522	516	510	504	498
492	487	481	475	469	464	458	452	446	441	435	430
424	418	413	407	402	396	391	386	381	375	370	365
360	355	350	344	339	334	330	325	320	315	310	306
301	296	292	287	283	278	274	269	265	261	256	252
248	244	240	236	232	228	224	220	216	213	209	205
201	198	194	191	187	184	181	177	174	171	168	164
161	158	155	152	149	146	144	141	138	135	133	130
127	125	122	120	117	115	113	110	108	106	013	101
99	97	95	93	91	89	87	85	83	81	80	78
76	74	73	71	69	68	66	65	63	62	60	59
57	56	55	53	52	51	50	49	47	46	45	44
43	42	41	40	39	38	37	36	35	34	33	32
31	31	30	29	28	28	27	26	25	25	24	23
23	22	22	21	20	20	19	19	18	18	17	17
16	16	15	15	14	14	14	13	13	13	12	12

(GAUSSIAN CURVE DATA BUFFER CONT.)

```
11   10   0    0    0    0    0    0    0    0    0    32
0    89   32   8    89   32   303  89   32   311  89   32
303  177  32   311  177  48   0    89   32   8    89   49
303  89   53   311  89   51   303  177  48   311  177  51
0    89   48   8    89   52   303  89   53   311  89   54
303  177  48   311  177
```

REFERENCES

1. Gilbert, D. D.; Mounts, T. T.; Frost, A. A. Journal of Chemical Education. 1982, 59, vol 8, 661.

2. Scanlon, Leo J., Forth Programming, Howard W. Sams & Co. Inc. Indianapolis, Indiana, 1982, pp 11-21.

3. Brodie, Leo, Starting Forth, Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1981, pp 1-27.

4. Skoog, Douglas A.; West,Donald M.Fundamentals of Analytical Chemistry, Holt, Rinehart, and Winston, New York, 1976, pp 54-58.

5. McNair, H. M.; Bonelli, E. J., Basic Gas Chromatography, Varian, Palo Alto, California, 1968, pp 123-134.

8. McReynolds, W. O. Gas Chromatographic Retention Data, Preston Publications, Inc. Niles, Illinois, 1966, pg 21.

7. Martin, A. J.; Synge,R.L. J. Chromatography, 1959, 2, 833.

8. Norgre, S. D.; Juvet, R. S. Gas Liquid Chromatography Theroy and Practice, John Wiley & Sons, New York, 1963.

9. McReynolds, W. O. Gas Chromatographic Retention Data, Preston Publications, Inc. Niles, Illinois, 1966, pp 3-21.

10. Dean, John A. Chemical Separations Methods, Van Vostrand Reinhold Comp. New York, 1969, pp 230-233.

11. McReynolds, W. O. Gas Chromatographic Retention Data, Preston Publications, Inc. Niles, Illinois, 1966.

12. Miller James M. Separation Methods in Chemical Analysis John Wiley & Sons, New York, 1975, pp 301-302.

13. Martin, A. J.; Synge, R. L., Biochem, J. 1941, 35, 1358.

14. Skoog, Douglas A.; West,Donald M.Fundamentals of Analytical Chemistry, Holt, Rinehart, and Winston, New York, 1976.

15. Norgre, S. D.; Juvet, R. S. Gas Liquid Chromatography Theroy and Practice, John Wiley & Sons, New York, 1963, pg 63.